Evaluación automática de test neuropsicológicos utilizando técnicas de graph matching

Realizado por:
Roberto Cerrillo Ayuso

Dirigido por:
Mariano Rincón Zamorano

Quiero agradecer a mi tutor Mariano Rincón y a mi familia por todo el apoyo mostrado durante el desarrollo de este trabajo.

Índice

1. Introducción 12

1.1. Motivación 12

1.2. Trabajos previos de partida 14

1.3. Objetivo e hipótesis de investigación 15

1.4. Metodología e hipótesis de investigación 16

1.5. Descripción del dataset empleado 17

1.5.1. Test neuropsicológicos utilizados en el proyecto 18

1.5.2. Bucles 18

1.5.3. Casa 19

1.5.4. Círculo 19

1.5.5. Cruz 20

1.5.6. Cuadrado 20

1.5.7. Cubo 21

1.5.8. Minimental 21

1.5.9. Picos-mesetas 22

1.5.10. Triángulo 22

2. Revisión de la bibliografía 23

2.1. Trabajos previos en el análisis de los dibujos en test neuropsicológicos . . . 23

2.2. Técnicas de reconocimiento de formas en imágenes y su aplicación al problema 27

2.2.1. Descriptores de formas basados en contornos 29

2.2.1.1. Shape context 29

2.2.1.2. Modelo de contornos fragmentados o “Boundary Fragment Model” 29

2.2.1.3. k-Segmento Adyacente o “k-Adjacent Segment” (kAS) . . . 30

2.2.1.4. Espacio de escala de curvatura o "Curvature scale space" (CSS) 30

2.2.1.5. Turning function 31

2.2.2. Descriptores de formas basados en regiones 32

2.2.3. Descriptores de formas basados en puntos 33

2.3. Uso de técnicas de graph matching en problemas de reconocimientos de patrones en imágenes 34

3. Descripción de la solución 36

3.1. Preprocesamiento de las imágenes 36

3.1.1. Planteamiento 36

3.1.2. Implementación 37

3.2. Creación de los grafos 39

3.2.1. Planteamiento 39

3.2.2. Teoría 39

3.2.3. Implementación 40

3.2.3.1. Aproximación de los contornos de una figura 40

3.2.3.2. Simplificación de vértices y creación del grafo base 44

3.2.3.3. Cálculo de los atributos de los nodos 46

3.3. Cálculo de la similitud entre el modelo y el dibujo 49

3.3.1. Planteamiento 49

3.3.2. Teoría - Similitud de grafos 49

3.3.3. Implementación 55

3.3.3.1. Cómputo de la matriz de coste de los nodos 56

3.3.3.2. Cómputo de la matriz de coste de los enlaces en la sustitución de nodos 60

3.3.3.3. Grado de similitud y asignación de la correspondencia entre los nodos de g1 y g2 61

3.3.4. Implementación de la evaluación de las asignación de GED 62

3.4. Métricas para el diagnóstico del DCL 64

4. Resultados 66

4.1. Resultados del preprocesamiento de las imágenes 66

4.1.1. Conclusiones 69

4.2. Resultados de las aproximaciones de contornos 69

4.2.1. Conclusiones 71

4.3. Resultados de la evaluación de las asignaciones 72

4.3.1. Resultados de las evaluaciones para la aproximación con esquinas Harris 73

4.3.1.1. Mejores configuraciones 75

4.3.2. Resultados de las evaluaciones para la aproximación con RDP . . . 76

4.3.2.1. Mejores configuraciones 78

4.3.3. Conclusiones 78

4.4. Resultados del diagnóstico del DCL 81

5. Conclusiones 85

5.1. Respuestas a las hipótesis 85

5.2. Conclusiones finales 85

Índice de figuras

1. Ejemplos de dibujos utilizados en los test neuropsicológicos objetos de este trabajo. 13

2. Criterios de corrección para las figuras de picos-mesetas y bucles. 14

3. Ejemplos de bucles por tipos de pacientes 18

4. Ejemplos de casas por tipos de pacientes 19

5. Ejemplos del círculo por tipos de pacientes 19

6. Ejemplos de la cruz por tipos de pacientes 20

7. Ejemplos del cuadrado por tipos de pacientes 20

8. Ejemplos de cubos por tipos de pacientes 21

9. Ejemplos del minimental por tipos de pacientes 21

10. Ejemplos de picos-mesetas por tipos de pacientes 22

11. Ejemplos del triangulo por tipos de pacientes 22

12. Aplicación de WDIH en una subregión del test del reloj. 23

13. Ejemplo de distorsiones del test del reloj 24

14. (izquierda) - Secciones de puntuación elegidas en el artículo [11]. (derecha) - Ejemplos de de secciones localizadas. 25

15. Extracción de lineas horizontales , verticales y diagonales de los 6 dibujos de la izquierda. 27

16. Clasificación de las técnicas descriptoras de formas según establece Anh Phuong atendiendo a la fuente de la que extraen las características de la formas. 28

17. Clasificación de las técnicas descriptoras de formas según establece Anh Phuong atendiendo a si el emparejamiento de las formas se hace utilizando características globales o locales. 28

18. (a) Ejemplo del contexto de forma de un punto (b) Contextos de forma de 3 puntos. 29

19. Ejemplo de descriptores CSS 31

20. Cálculo de distancia de similitud para dos turning function. 32

21. [30]. 33

22. Ejemplos de dibujos originales con trazos muy débiles. 36

23. Ejemplo de las diferencias de tamaño entre dos sesiones realizadas por un mismo paciente. 37

24. Proceso de traslación y escalado de una imagen del paciente 38

25. Ejemplos de la simplificación de los contornos. 44

26. Ejemplos de la simplificación de los contornos de los círculos. 44

27. Ejemplos de de las etapas de creación del grafo. 46

28. Ejemplo del atributo Vector_CM para la imagen del cuadrado modelo. . . 47

29. Ejemplo del atributo Vector_Suma para un nodo de la imagen del cuadrado modelo. 47

30. Ejemplo del atributo Ángulos_nodo para un nodo de la imagen del cuadrado modelo. 48

31. Ejemplos de exact graph Matching (izquierda) y de inexact graph matching (derecha). 49

32. Ejemplo del camino de ediciones que transforma el grafo g_1 en el grafo g_2. . 50

33. Ejemplo del cálculo del coste del atributo Ángulos_nodo para el mismo nodo de dos cubos similares. 58

34. Etiquetación de todos los modelos. 63

35. Etiquetación de una casa en la que faltan bastantes regiones. 63

36. Etiquetación de una cruz en la que faltan algún punto. 64

37. Ejemplos de imágenes preprocesadas (original izquierda y preprocesada a la derecha) con ruido de digitalización. 66

38. Umbralización de Otsu con dos y tres niveles para un dibujo con trazos débiles. Se ha indicado con círculos rojos las zonas con trazos débiles que mejor han sido digitalizadas por la de tres niveles en comparación con la de dos. 67

39. Ejemplos de imágenes preprocesadas (original izquierda y preprocesada a la derecha) con trazos débiles. 68

40. Ejemplo de transformaciones realizadas en el preprocesamiento de las imágenes con poca cantidad de ruido o sin trazos débiles. 69

41. Ejemplos de la simplificación de los contornos de los círculos. 72

42. Ejemplo de asignaciones (1) devueltas por GED (grafos modelos arriba y grafos de pacientes abajo). En rojo se indican los nodos de g1 que han sido eliminados y en verde los nodos que se han insertado en g2. 79

43. Ejemplo de asignaciones (2) devueltas por GED (grafos modelos arriba y grafos de pacientes abajo). En rojo se indican los nodos de g1 que han sido eliminados y en verde los nodos que se han insertado en g2. 80

44. Gráfica con los resultados del diagnóstico para la aproximación por esquinas Harris 84

45. Gráfica con los resultados del diagnóstico para la aproximación por RDP . 84

Algoritmos

1. Algoritmo de aproximación de contornos utilizando las esquinas de Harris. 41
2. Algoritmo de distancia de edición de grafos A^* mostrado en [39] 51
3. Algoritmo de Munkres para el problema de asignación 53
4. Algoritmo de cálculo de la matriz de coste de sustitución de nodos para el atributo Ángulos_nodo. 58

Índice de cuadros

1. Descripción del dataset empleado en este proyecto 17

2. Resultados de la evaluación de las aproximaciones con esquinas Harris y RDP según el tipo de dibujo y de paciente. 70

3. Resultados de la evaluación de las aproximaciones con esquinas Harris y RDP según el tipo de dibujo. 71

4. Resultados de la evaluación de las asignaciones de GED para la aproximación con esquinas Harris utilizando únicamente una matriz de coste 74

5. Resultados de la evaluación de las asignaciones de GED para la aproximación con esquinas Harris combinando varias matrices de matriz de coste . . 75

6. Resultados de la evaluación de las asignaciones de GED para la aproximación RDP utilizando únicamente una matriz de coste 77

7. Resultados de la evaluación de las asignaciones de GED para la aproximación con RDP combinando varias matrices de matriz de coste 77

8. Comparativa de la evaluación de las asignaciones de GED para la aproximación con esquinas Harris y RDP. 78

9. Diagnóstico DCL para la aproximación con esquinas Harris 82

10. Diagnóstico DCL para la aproximación con RDP 83

Resumen

Este proyecto propone una metodología para evaluar automáticamente la parte gráfica de varios test neuropsicológicos que los psicólogos emplean para medir la severidad del deterioro cognitivo leve. Los test consisten en la copia y reproducción, a mano alzada, de un dibujo modelo por parte del paciente, en dónde, el grado de deterioro cognitivo leve se mide evaluando las diferencias del dibujo con el modelo. El método propuesto parte de desarrollos anteriores en los que se trabajó sobre un dibujo en particular, por una parte, y sobre un conjunto de ellos a través del análisis de las características globales de los dibujos. En este trabajo se propone un método genérico al tipo de dibujo basado en modelizar, lo más fidedignamente posible, los elementos que componen los dibujos así como sus relaciones internas. En particular se van a emplear técnicas de graph matching para medir la semejanza entre los dibujos y recoger las variabilidad de distorsiones. Y de esta manera extraer unas métricas que cuantifiquen las diferencias de los distintos tipos de deterioro cognitivo leve.

Palabras Clave: Dibujos en test neuropsicológicos, Deterioro Cognitivo Leve, Análisis de dibujos, Semejanza entre dibujos, Graph Matching, Pattern Matching.

1. Introducción

1.1. Motivación

El deterioro cognitivo leve (DCL) es una situación clínica en la cual los déficit en las funciones cognitivas son claros pero no lo suficientemente graves para el diagnóstico de la demencia. Los individuos afectados presentan daños más allá de lo esperado para su edad pero que no representa perjuicio significativo en sus actividades diarias. En la mayoría de los casos, representa un estado de transición entre el envejecimiento normal y la demencia leve, generalmente como enfermedad de Alzheimer, aunque también deriva hacia otros tipos de demencia. Aunque en una pequeña parte de los individuos afectados no hay evolución e incluso puede remitir. Actualmente no existe un tratamiento farmacológico que permita revertir el proceso por lo que es de gran importancia un diagnóstico precoz y un seguimiento. De esta manera se pueden aplicar tratamientos que puedan mejorar los síntomas o prevenir o retrasar el avance de la demencia. El diagnóstico del DCL se hace en el entorno clínico y se basa en varios tipos de pruebas de distinta índole como: la realización de exámenes neurológicos, análisis de sangre, análisis de imágenes cerebrales o la realización de un estudio del estado mental utilizando test neuropsicológicos. Y es la evaluación automática de varios test neuropsicológicos en donde se enmarca el contexto de este proyecto como voy a exponer a continuación.

El abanico de test neuropsicológicos que emplean los médicos y psicólogos en el diagnóstico del DCL es muy amplio ya que miden distintas capacidades intelectuales como: la capacidad verbal, la atención, la memoria, las funciones ejecutivas, las funciones espaciales y constructivas o las praxias y gnosias. La elección de un test u otro, o la realización de otro tipo de pruebas, depende del clínico. Es él quién tiene que elegir en función de su disponibilidad y contexto que prueba realizar tal y como expone Claver-Martín en [1].

Los test de interés en este proyecto son aquellos que evalúan la reducción de la capacidad de reproducir o copiar figuras simples. El conjunto de test provienen de: el El Mini Examen Cognoscitivo (MEC) [2], el Test Barcelona [3] y la figura compleja de Rey [4]. A continuación, figura 1, muestro un ejemplo de cada test al igual que indico el nombre asociado a cada figura y que va ser el utilizado a lo largo de esta memoria.

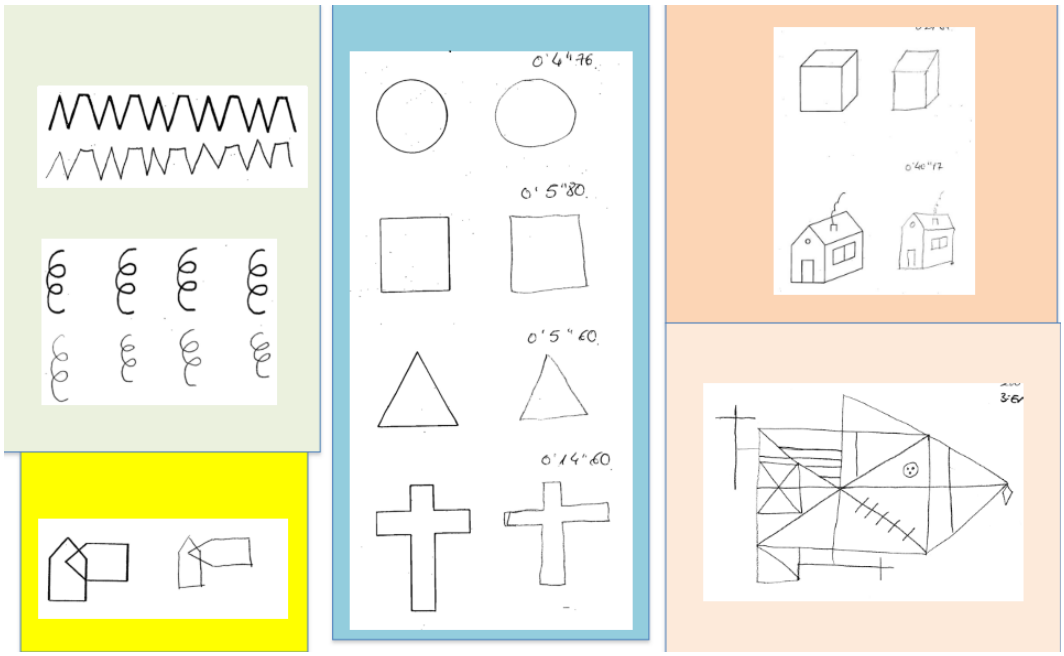


Fig. 1: Ejemplos de dibujos utilizados en los test neuropsicológicos objetos de este trabajo. La figura de minimental (recuadro amarillo) pertenece al MEC [2]. La figuras del Test Barcelona [3] son: picos-mesetas (arriba a la izquierda), bucles (la del centro a la izquierda), el círculo, el cuadrado, el triangulo, la cruz, el cubo y la casa. La figura compleja de Rey es la de abajo a la derecha [4].

Independientemente del tipo de test, al paciente se le pide que copie o reproduzca una figura modelo o patrón. A partir de esta ejecución el personal clínico evalúa el grado de distorsión del dibujo. Hay que destacar que la variabilidad de distorsiones es muy amplia como por ejemplo: la falta de elementos del modelo, o cambios muy apreciables de escala y rotación, o el añadido de elementos, o dibujos con trazos curvos en vez de rectilíneos, etc.

Un aspecto importante de la evaluación de estos test es que hay cierta subjetividad al puntuarlos a pesar de la existencia de un criterio de puntuación. Lo que puede llevar a discrepancias en el diagnóstico si no se corrigen las discrepancias entre los evaluadores [5]. Por ejemplo, en la siguiente imagen muestro los criterios de corrección para las figuras de picos-mesetas y bucles. Podemos ver como únicamente se puntúa cada ítem con 0,1 y 2. Por ejemplo una puntuación de 0 se refiere a «ítem mal realizado. Defectos claros en la seriación de los movimientos. Contaminaciones. Perseveraciones. Ausencia de respuesta. Closing con el modelo». Aquí, por ejemplo, entrarían bucles rotados o a los que le falta algún bucle o tienen alguno demás.

ALTERNANCIA GRÁFICA Y BUCLES		*VALORACIÓN*			
		Correcta	Ejecución regular	Incorrecta	Total
Lámina 1	**Picos-Mesetas**	2	1	0	
	Bucles	2	1	0	
Totales					

Puntuación:

- 2 puntos por ítem realizado correctamente. Rápido, sin correcciones.

- 1 punto por ítem realizado regularmente. La reproducción es la solicitada pero con correcciones, lentificación significativa… pérdida discreta del ritmo.

- 0 puntos por ítem mal realizado. Defectos claros en la seriación de los movimientos. Contaminaciones. Perseveraciones. Ausencia de respuesta. Closing con el modelo.

Fig. 2: Criterios de corrección para las figuras de picos-mesetas y bucles.

El caso de los bucles mostrado en el párrafo anterior sucede en todas las figuras. Por lo que es muy interesante desarrollar, y este es el **propósito de este trabajo**, una evaluación automática que caracterice adecuadamente las distorsiones y de esta manera complementar la evaluación dada por el personal clínico. Por ejemplo el método podría mostrar distorsiones que son significativas de un cierto grado del deterioro cognitivo leve.

1.2. Trabajos previos de partida

Este trabajo prosigue la línea de investigación del grupo SIMDA [1] sobre la evaluación automática de los test de reproducción y copia de figuras. Concretamente continua el trabajo previo efectuado por M. Rincón, S. García-Herranz, M. C. Díaz-Mardomingo, R. Martínez-Tomás, H. Peraita en [8]. A su vez, este TFM también es continuación del trabajo en grupo que desarrollé en colaboración con Mikel Val Calvo y Patricia Mayo Tejedo [9].

En [8] se propone un método para evaluar el test de los picos-mesetas (dibujo de arriba a la izquierda en la figura 1). El método es capaz de generar varías métricas (segmentos válidos, factor de escala del eje X, y distancia vertical al modelo) que caracterizan adecuadamente los dibujos y que son robustas e independientes de la evaluación del clínico. Consiguen buenos resultados discriminando entre pacientes sanos y pacientes con cierto grado de DCL. Para ser más exactos consiguen clasificar los dos grupos con una precisión de 0.777 y una sensibilidad de 0.771. El único problemas es que la implementación para la figura de picos-mesetas es ad-hoc. Y esto lleva a que no sea fácilmente reutilizable en el resto de figuras. Por este motivo, es interesante desarrollar un método, que aplicado de una manera

[1] `http://simda.uned.es/`

genérica, combine las características del resto de figuras y que mejore los resultados de la clasificación.

En [9] desarrollamos un método general para el análisis automático sobre la mayoría de los dibujos mostrados en la figura 1 (a excepción de la figura del rey de abajo a la derecha). Con este método obtuvimos una precisión aproximada del 60 % para las imágenes complejas (casa, bucles). En cambio, con las figuras simples como el triángulo, el cuadrado o el círculo obtuvimos precisiones cercanas al 70 %. Y precisiones en el torno del 80 % para la cruz y los picos-mesetas.

El método desarrollado en [9] calcula una serie de métricas globales para cada figura del paciente. En cada una de ellas se extrae un conjunto de contornos ordenados según el perímetro y según si un contorno se encuentra en el interior de otro más grande. Posteriormente se aplica una etapa de reducción y seguidamente se calcula la firma de cada contorno (aplicando "Turning Functions"). El método termina devolviendo un vector de n componentes (n igual al número de contornos de cada figura) . Cada uno ellos está formado por otro vector con las métricas globales de cada contorno (área, perímetro, orientación, firma del contorno, número de vértices, etc).

1.3. Objetivo e hipótesis de investigación

En las conclusiones de [8] se expone que en futuros trabajos se combinarán las características del resto de figuras para mejorar el diagnóstico de DCL, e igualmente, en las conclusiones de [9] se comenta que es necesario realizar análisis futuros que tengan en cuenta la relación entre los elementos que conforman. El uso de grafos y de técnicas de emparejamiento de grafos o "graph matching" (GM) es una de las maneras de registrar las relaciones internas, a la par que proporcionar una medida de similitud. Tal y como mostraré en este proyecto considero que las técnicas de graph matching pueden ser adecuadas para ambos propósitos. Y además, de poder registrar, con independencia de la forma de la figura, la gran variabilidad de distorsiones.

Llegado a este punto, y teniendo en cuenta las consideraciones del párrafo anterior, expongo el **objetivo** de este trabajo:

Analizar, con técnicas de "graph matching", la información estructural entre el dibujo del paciente y su patrón para cuantificar las diferencias y así proporcionar un diagnóstico del DCL.

A partir del objetivo anterior, establezco dos **hipótesis generales** que considero necesarias para la consecución del mismo.

1. Las diferencias entre el dibujo del paciente y su patrón determinan el grado de DCL. **Por lo que, cuanto mayor sean las diferencias mayor será el DCL.**

2. El nivel de registro de los trazos de los dibujos influye en la capacidad discriminatoria de la cuantificación de las diferencias. **Es decir, cuanto más se asemeja el modelo al dibujo de partida, mejor será el diagnóstico del DCL.**

El empleo de técnicas de graph matching implica trasladar el empleo de los grafos alas dos hipótesis generales el empleo de los grafos. De esta manera establezco dos **hipótesis particulares** relacionadas directamente con las anteriores:

1. Graph matching calcula un valor de las diferencias (y otras métricas) entre el grafo patrón y el del dibujo del paciente, al igual que una asignación entre vértices del dibujo patrón y del paciente. **Cuanto mejor sea la asignación mayor será el poder discriminatorio de las métricas del DCL.**
2. Cada grafo debe tener un aspecto lo más similar posible al aspecto del dibujo. Es decir, las formas, orientaciones y longitudes de los segmentos del dibujo se tiene que respetar en el grafo.

1.4. Metodología e hipótesis de investigación

Tanto la metodología propuesta en [8] y en [9] son un buen punto de partida para desarrollar la mía debido a que el propósito de este trabajo es el mismo que el de ambos. Además trabajo con las misma imágenes. Generalizado, concreto que cualquier propuesta de solución del problema debe cubrir las siguientes etapas:

1. Generación de un modelo computable de las figuras
2. Extracción de un conjunto de características (empleando técnicas de reconocimiento de patrones) que caractericen la comparación entre cada par dibujo-patrón.
3. Diagnóstico DCL.

Considero la generación del modelo cómo la etapa más importante para el devenir de la solución debido a la importancia de generar un modelo capaz de registrar adecuadamente las relaciones internas de las líneas que componen el dibujo incluyendo el registro de las distorsiones que pueda contener. En general, esta etapa engloba las fases de bajo y medio nivel de una aplicación de visión artificial. Es decir aquellas fases que trabajan con la información de bajo y medio nivel de la imagen. En el procesamiento a bajo nivel se adquiere la imagen original y se realiza un preprocesamiento para obtener una imagen adecuada al ámbito del problema. Y en el de medio nivel se segmenta la imagen en áreas con significado para posteriormente trabajar con estas áreas y generar una descripción de la imagen. Resumiendo estas fases son:

1. Adquisición de la imagen original
2. Preprocesamiento
3. Segmentación
4. Representación y descripción

En cuanto a mi proyecto creo que es suficiente el uso de las técnicas habituales de bajo nivel y es en cuanto a las de medio nivel donde voy a tener que poner más énfasis debido a que los grafos van a representar y describir cada figura (incluyendo las distorsiones). Partiendo de las hipótesis particulares 1.3 establezco varias **condiciones**:

1. Para que la asignación obtenida del graph matching sea verosímil hay que evaluar las correspondencias generadas a través de una comprobación supervisada (hipótesis particular 1).
2. El método de creación de grafos tiene que ser independiente de la figura tratada.

El diagnóstico del DCL se va a realizar evaluando, con comparadores, las métricas proporcionadas por el proceso de graph matching para discernir entre individuos sanos y pacientes con DCL.

1.5. Descripción del dataset empleado

El dataset empleado es el mismo que el utilizado en [8]. Se compone de una muestra de 40 participantes recogida de un estudio longitudinal durante 3 años (ref. SEJ 2004-04233 y SEJ 200763325) y que se centró en determinar la prevalencia de los subtipos de DCL. Se establecieron tres categorías de DCL en función de la puntuación obtenida en los test. A 16 pacientes se les clasificó como sanos al tener puntuaciones que entraban dentro de las escalas de referencia. 24 paciente fueron clasificados con DCL al encontrarse su puntuación 1.5 DS (desviación estándar) por debajo de la media en al menos dos de los test. De los 24 pacientes diagnosticados con DCL, 10 fueron clasificados como DCL Estable al diagnosticarles síntomas probables de una fase inicial de enfermedad de Alzheimer. El resto de los 24 pacientes con DCL son categorizados como DCL Evolución. En la siguiente tabla se muesta un resumen de las características de los pacientes y sus puntuaciones.

Tab. 1: Descripción del dataset empleado en este proyecto

	Sanos n=16	**DCL Estable n=10**	**DCL Evolución n=14**
	Media (DS)	Media (DS)	Media (DS)
Género (femenino)	13 (81.25 %)	11.00 (4.24)	4 (40 %)
Edad (años)	69.75 (4.85)	72.57 (5.31)	71.60 (5.08)
Educación formal (años)	11.00 (4.24)	5.50 (6.51)	8.30 (5.53)
Escala de Depresión Geriátrica (GDS) (escala de Yesavage)	3.44 (2.78)	3.93 (2.99)	4.20 (3.39)
Nivel Funcional (escala de Blessed)	0.59 (0.45)	0.67 (0.57)	1.00 (1.00)
Estado cognitivo (MEC 0-35)	33.50 (2.47)	29.14 (3.78)	30.30 (2.00)

1.5.1. Test neuropsicológicos utilizados en el proyecto

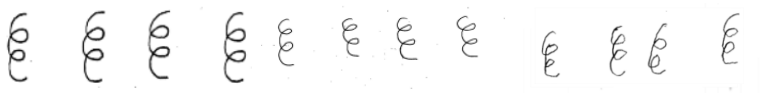
Seguidamente se exponen parte de los test neuropsicológicos empleados en el estudio y que forman el conjunto de imágenes utilizadas en este proyecto.

1.5.2. Bucles

Los bucles forman parte del conjunto de pruebas de alternancia gráfica y bucles (Peña-Casanova, 1991). Evalúan algunos componentes de la función ejecutiva como la capacidad de cambio, shift, planificación y secuenciación. En su análisis se observa que los dos rizos del bucle sean del mismo tamaño. Además se analiza el ritmo al realizar los cuatro bucles. En pacientes con cierto grado de DCL se detectan dificultades en hacer los dos bucles iguales y no hay homogeneidad en las formas de los cuatro bucles.

Fig. 3: Ejemplos de bucles por tipos de pacientes

(a) Modelo (b) Paciente sano (c) Paciente con DCL



1.5.3. Casa

La casa pertenece al conjunto de formas diseñadas por Peña Casanova para medir las praxias constructivas de los individuos. Pacientes con DCL tienen dificultades a la hora de crear profundidad en las figuras tridimensionales. Ya sea por no reflejar adecuadamente la perspectiva o por ubicar secciones de la casa en otras posiciones.

Fig. 4: Ejemplos de casas por tipos de pacientes

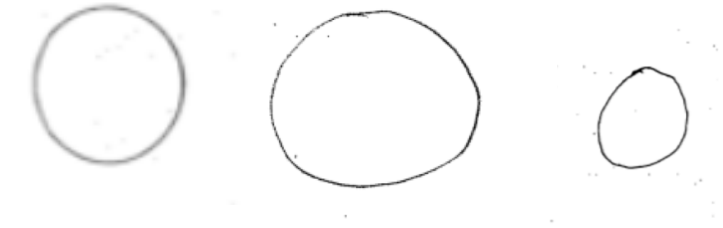
(a) Modelo (b) Paciente Sano (c) Paciente con DCL

1.5.4. Círculo

Pertenece al conjunto de formas diseñadas por Peña Casanova para medir las praxias constructivas de los individuos. En pacientes con DCL se aprecian que el tamaño del círculo es inferior al modelo o realizan círculos más ovalados o con cambios en la orientación.

Fig. 5: Ejemplos del círculo por tipos de pacientes

(a) Modelo (b) Paciente Sano (c) Paciente con DCL

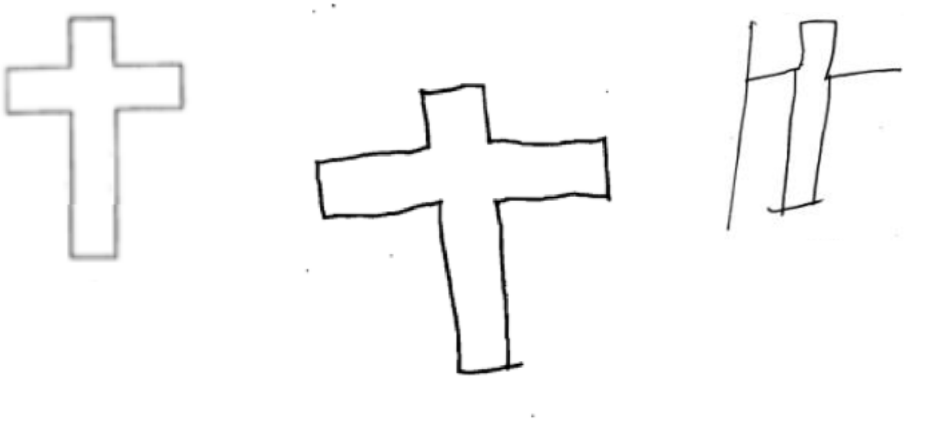


1.5.5. Cruz

La cruz pertenece al conjunto de formas diseñadas por Peña Casanova para medir las praxias constructivas de los individuos, en el subgrupo de las figuras de dos dimensiones. Entre las distorsiones más habituales en pacientes con DCL se encuentran trazos curvos, o trazos añadidos.

Fig. 6: Ejemplos de la cruz por tipos de pacientes

(a) Modelo (b) Paciente Sano (c) Paciente con DCL

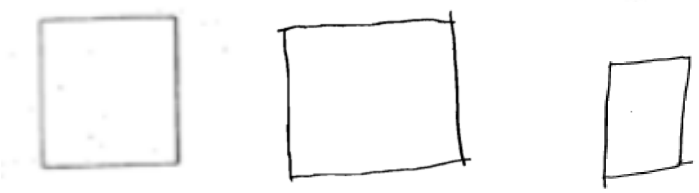


1.5.6. Cuadrado

El cubo pertenece al conjunto de formas diseñadas por Peña Casanova para medir las praxias constructivas de los individuos, en el subgrupo de las figuras de dos dimensiones. Los defectos habituales son las diferencias de tamaño. Pacientes con DCL habitualmente realizan rectángulos pequeños.

Fig. 7: Ejemplos del cuadrado por tipos de pacientes

(a) Modelo (b) Paciente Sano (c) Paciente con DCL

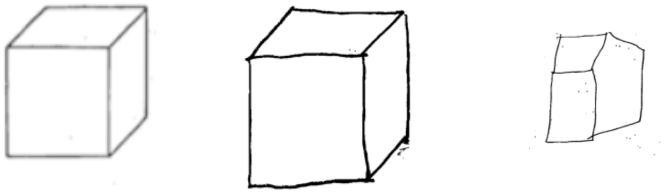


1.5.7. Cubo

Pertenece al conjunto de formas diseñadas por Peña Casanova para medir las praxias constructivas de los individuos, en el subgrupo de las figuras más complejas tridimensionales. Se nota que los pacientes con problemas cognitivos tienen dificultad a la hora de crear profundidad en las figuras tridimensionales, dibujando líneas de más o de menos. También se encuentran problemas con el tamaño y los ángulos entre líneas.

Fig. 8: Ejemplos de cubos por tipos de pacientes

(a) Modelo (b) Paciente Sano (c) Paciente con DCL

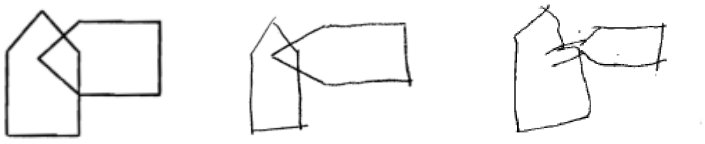


1.5.8. Minimental

El minimental forma parte del examen cognoscitivo MEC de Lobo y se evalúa comprobando que las dos figuras se cortan por dos puntos que es precisamente uno de los errores comunes entre pacientes enfermos. Se encuentran también fallos en el tamaño de las figuras, su posición relativa entre ellas y su orientación.

Fig. 9: Ejemplos del minimental por tipos de pacientes

(a) Modelo (b) Paciente Sano (c) Paciente con DCL



1.5.9. Picos-mesetas

Al igual que los bucles pertenecen a las pruebas alternancia gráfica y bucles (Peña-Casanova, 1991). Pacientes con DCL no siguen el ritmo adecuadamente, por lo que la figura tiende a acortarse o alargarse, a subirse para arriba o para abajo.

Fig. 10: Ejemplos de picos-mesetas por tipos de pacientes

(a) Modelo (b) Paciente Sano (c) Paciente con DCL

1.5.10. Triángulo

El triangulo pertenece al conjunto de formas diseñadas por Peña Casanova para medir las praxias constructivas de los individuos, en el subgrupo de las figuras de dos dimensiones. Entre las distorsiones más habituales en pacientes con DCL hay diferencias de forma, tamaño o trazos quebrados.

Fig. 11: Ejemplos del triangulo por tipos de pacientes

(a) Modelo (b) Paciente Sano (c) Paciente con DCL

2. Revisión de la bibliografía

2.1. Trabajos previos en el análisis de los dibujos en test neuropsicológicos

Han sido varios los trabajos previos desarrollados que abordan el problema de diagnóstico de pacientes con Alzheimer o DCL u otro trastorno cognitivo, a través de los test con dibujos a mano alzada. En [10] se propone un método para la extracción de características en el test del reloj para poder distinguir entre pacientes sin deterioro cognitivo y pacientes con cierto tipo de demencia. Este método utiliza "Weighted Direction Index Histogram" para el modelado del contorno como podemos observa en la figura 2. Nos devuelve un histograma de cada subregión de la imagen, el cual refleja la dirección y módulo del contorno en esa subregión. Este histograma es reducido a un vector de características utilizando un filtro de pesos espaciales basado en una distribución gausiana. Y es este vector a partir de cual se hacen las comparaciones entre el dibujo y el patrón. Para realizar esta comparación utilizan la función "Modified Bayesian Discriminant Function" (MBDF). La cual devuelve las discordancias entre la imagen del dibujo y el patrón del reloj.

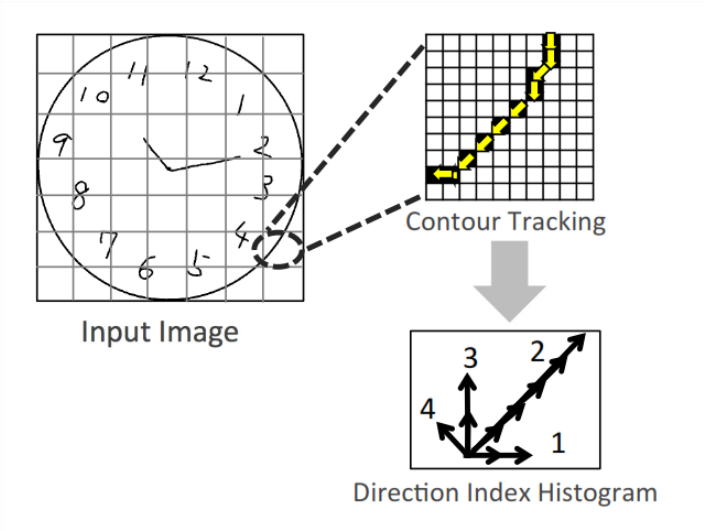


Fig. 12: Aplicación de WDIH en una subregión del test del reloj.
Ilustración extraída del artículo [10].

Exponen que han obtenido una alta especificad y sensibilidad (0.94 y 0.82 respectivamente) en la clasificación de individuos sanos e individuos con demencia. En cambio también comentan que se clasificaron incorrectamente imágenes de pacientes sanos debido a las diferencias en tamaño de los dígitos y las formas de las manecillas del reloj (test de la parte superior de la figura 3). También comentan que hay dibujos de pacientes con demencia que fueron clasificados como individuos sanos (test de la parte inferior de la figura 3), aunque

tal y como podemos apreciar, incluso para un especialista parece complicado distinguir la demencia en los dibujos debido a que no hay diferencias significativas que los diferencien de los sanos. Continúan diciendo que para poder reconocerlos adecuadamente se requiere más información, como por ejemplo un análisis más fino de los trazos.

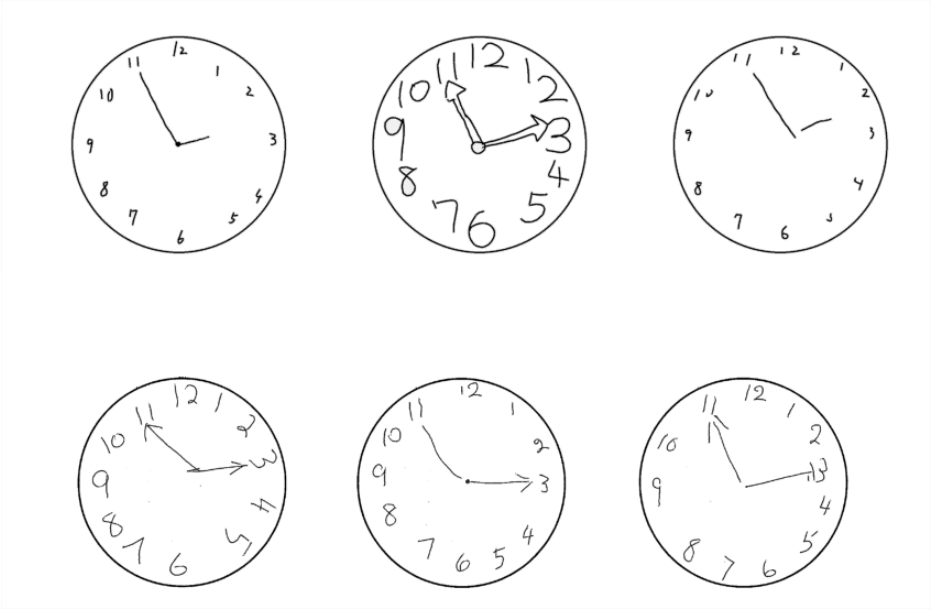


Fig. 13: Ejemplo de distorsiones del test del reloj
Los tres dibujos superiores son ejemplos de casos sanos pero con grades diferencias en la representación de los dígitos y manecillas . Y las tres de abajo representan test de pacientes con demencia que han sido clasificados como sanos. Ilustración extraída del artículo [10].

El artículo [11] desarrolla un método bastante válido para el grado de distorsión de la figura del rey. El cual se trata posiblemente del test más complejo dentro del abanico de todos los test neuropsicológicos debido a que contiene muchos polígonos y vértices internos, además de bastantes segmentos cortos. Este método considera una serie de distorsiones que no se suele tener en cuenta. Sin embargo sí que son importantes en el contexto de nuestro problema. Por ejemplo, considera que un segmento recto válido también es aquel que contienen curvas significativas o que está formado por líneas quebradas con cierta separación entre ellas. O igualmente aquellos segmentos que aún siendo rectos se desvían un cierto grado de la línea recta ideal. El sistema descompone la figura rey en 18 secciones puntuables, las cuáles son puntuadas basándose en su localización y grado de distorsión. Aunque en este trabajo sólo se puntúan las secciones rectangulares, triangulares romboidales de la figura 4 (imagen izquierda).

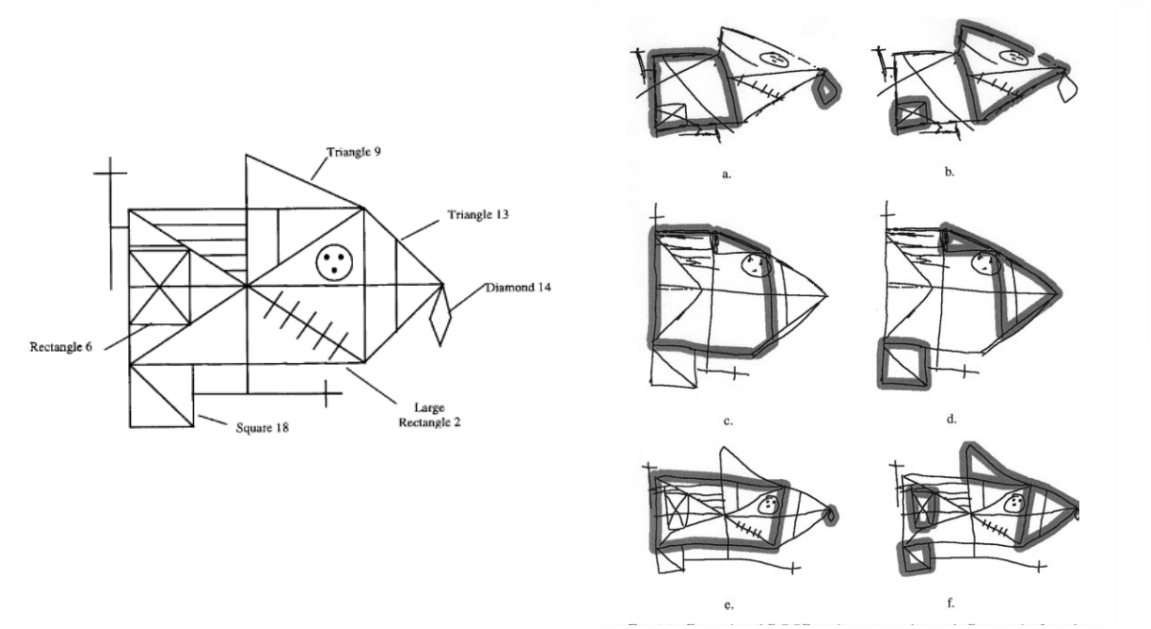


Fig. 14: (izquierda) - Secciones de puntuación elegidas en el artículo [11]. (derecha) - Ejemplos de de secciones localizadas.
El rectángulo 2 y el rombo 14 están localizados en a, c y e. El rectángulo 6, triángulo 9, triángulo 13 y cuadrado 18 están presentes en b, d y f.

Al principio se utiliza una técnica borrosa para agrupar en una sola linea segmentos que son colineales en naturaleza. A continuación se crea un grafo relacional a partir de las lineas colineales anteriores. El grafo contiene tantos nodos como lineas colineales se hayan formado previamente, esto da una media de 22 enlaces por nodo. Después realizan una búsqueda de cada forma patrón (triángulo, rectángulo y rombo) a ser reconocida dentro del grafo. Cada linea es considerada como punto de partida para la búsqueda recursiva donde todos los nodos conectados al nodo actual se prueban para ver si este constituye: o un vértice, o una continuación de un lado recto, o un vértice de corte en el caso de que la forma a buscar sea un rectángulos. Si el nodo conectado pasa el test, entonces este se convierte en el nodo actual y la búsqueda continua aplicándose una búsqueda en profundidad o "Depth First Search". El proceso nos devuelve una figura candidata cuando la búsqueda ha retornado al nodo inicial con el numero correcto de lados encontrados. Seguidamente se almacena esta figura candidata y de nuevo se realiza un proceso de backtracking para que de esta manera se haga una búsqueda exhaustiva del grafo.

El siguiente paso es la caracterización de cada figura candidata. Para ello se utilizan las siguiente métricas: colinealidad de linea, máxima desviación de un lado, corrección de vértices y simetría en el caso del rombo. Finalmente también realizan un proceso de agrupamiento que reduce el número de figuras candidatas; al reducir a una única ocurrencia figuras que son muy similares entre si. En la imagen de la derecha de la figura 4 mostramos algunos ejemplos de secciones de puntuación localizadas por el algoritmo. Cómo dijimos con anterioridad este método obtiene muy buenos resultados. Para validarlo lo aplicaron

a 140 figuras hechas por niños asistente, y todas menos una fueron reconocidas como la figura del rey. Igual de satisfactorio fue el grado de distorsión producido por su método. El 98.6 % de las figuras fueron puntuadas con un error menor del 10 % (el 78 % con un error menor del 5 %) en comparación con la puntuación subjetiva dada por los especialistas.

Otro artículo que aborda la problemática es la explicada en el artículo [12]. Se muestra un método para segmentar automáticamente la imagen de un dibujo en sus componentes horizontales, verticales y diagonales, y devolviendo su localización dentro del dibujo. La primera etapa consiste en evaluar los píxeles dentro del dibujo y establecer la dirección dominante de su entorno. Si de primeras no se puede establecer una dirección predominante, el algoritmo aumenta el tamaño de la ventana (pasando de una matriz 3x3 a una de 5x5) repitiéndose la evaluación de la dirección. Una vez que se han evaluado todos los píxeles, se crean tres matrices que contienen los píxeles que han sido asignados como horizontal, vertical o diagonal (un píxel solo puede estar asignado a un tipo de dirección). En la siguiente etapa analizan cada imagen direccional para encontrar aquellos píxeles conectados que han sido ignorados en la etapa anterior y de este conjunto pasan a buscar los vértices de cada segmento. Ahora buscan la posibilidad de unir segmentos cercanos por lo que aplican un algoritmo de seguimiento de caminos ("path-following algorithm") que establece la posible conectividad entre pares de puntos finales. Si es positiva entonces el segmento más corto es copiado de la imagen original a la imagen direccional para preservar el contorno original. Continúan con la formación de una lista de componentes conectados a través de un proceso de etiquetación utilizando la mascara de "8 neighbourhood" con la que recogen la información posicional de cada elemento: vértices arriba-izquierda y abajo-derecha del área ocupado por el componente, la localización del centroide y la longitud del componente.

El alcance del método propuesto en [12] termina con la extracción de tres métricas que les permiten cuantificar la calidad del dibujo en comparación con el modelo. Establecen las siguientes métricas para evaluar la figura del cubo efectuada por un conjunto de individuos: número de componentes omitidos, diferencia de longitud y diferencia espacial. Todas estas métricas se calculan comparando las distancias de cada componente del dibujo con su respectiva del modelo, si una componente del dibujo es omitida no se tienen en cuenta sus métricas. Concluyen diciendo que las métricas son lo suficientemente significativas para discriminar entre los individuos de control y los pacientes con algún tipo de demencia. Y que suponen una base desde la que pueden establecer una evaluación estandarizada de la evolución del paciente. En la figura 5 vemos como ha sido la extracción de test de la parte de la izquierda. Los test a, b y c pertenecen a la respuesta de control y los test d, e y f a la respuesta de pacientes con cierto grado de demencia.

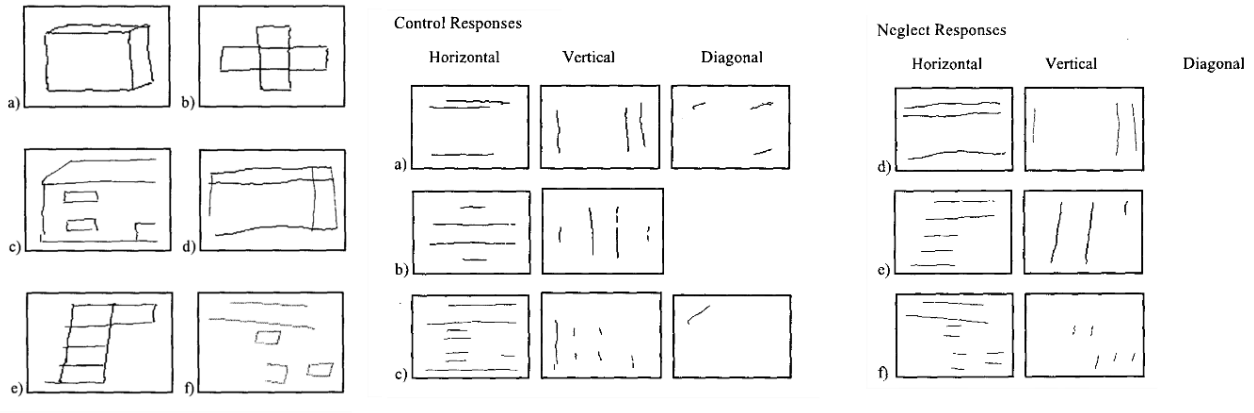


Fig. 15: Extracción de lineas horizontales , verticales y diagonales de los 6 dibujos de la izquierda.
Los test a, b y c pertenecen a la respuesta de control y los test d, e y f a la respuesta de pacientes con cierto grado de demencia. Ilustración extraída del artículo [12].

2.2. Técnicas de reconocimiento de formas en imágenes y su aplicación al problema

La literatura sobre el emparejamiento de formas o "Patern Matching" en imágenes es bastante amplia debido a que es un problema frecuente al que se enfrenta la visión artificial desde casi sus orígenes. Son técnicas que se utilizan en aplicaciones tales como: el reconocimiento facial, de texto manuscrito, o de huellas dactilares o por ejemplo como la detección de imágenes dentro de una colección o el procesamiento de imágenes biomédicas, vehículos autónomos, etc. Los métodos que se utilizan dependen en gran parte de las características intrínsecas de las imágenes que son capaces de procesar (color, textura, intensidad, etc.). Como hemos visto con anterioridad, las imágenes de los test neuropsicológicos son en escala de grises predominando el negro sobre fondo blanco. Por lo que la carencia de color o textura de los test limita bastante el uso de técnicas basadas en el color o la textura. Y por esto que estamos limitados a aplicar técnicas que describen las formas que constituyen una figura, llamadas descriptores de forma o "Shape Decriptors". A este problema también se enfrenta Anh Phuong en su tésis [15] en donde tiene que reconocer objetos en sketchs (bocetos o croquis) de guiones. En esta sección voy a centrarme en exponer las técnicas de formas.

Anh Phuong comenta que hay buenas recopilaciones de aproximaciones de formas[16] y que es habitual establecer dos categorías: descriptores basados en contornos y en regiones. En cambio prefiere añadir una tercera categoría para las técnicas basadas en puntos. En la siguiente imagen muestro las tres categorías según expone Anh Phuong.

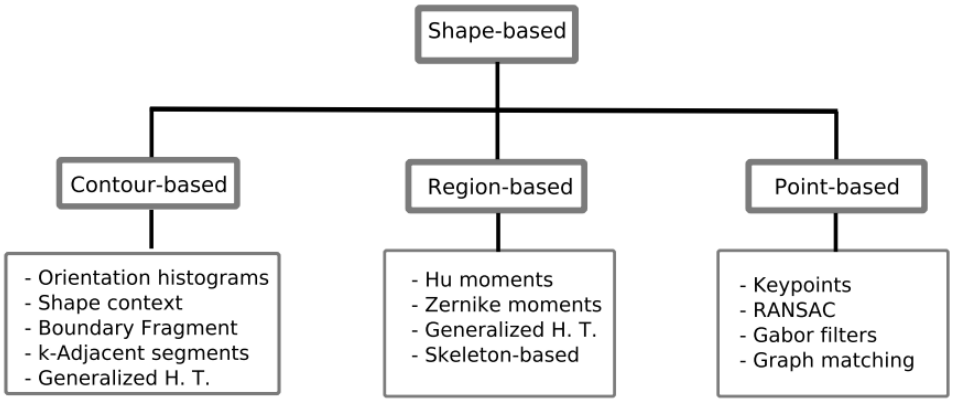


Fig. 16: Clasificación de las técnicas descriptoras de formas según establece Anh Phuong atendiendo a la fuente de la que extraen las características de la formas.
Ilustración extraída del artículo [15].

A su vez Anh Phuong describe otra clasificación de los descriptores de forma, pero esta vez basada en el uso de los métodos de emparejamiento, es decir, basada en si se utilizan características globales o locales para emparejar las figuras. De esta manera las técnicas anteriores se clasifican en dos grupos: emparejamientos basados en las características globales o en locales. Esta clasificación la expongo en la siguiente imagen.

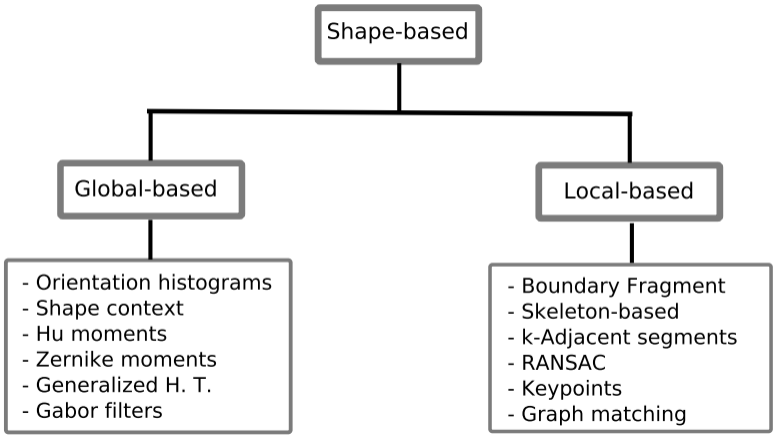


Fig. 17: Clasificación de las técnicas descriptoras de formas según establece Anh Phuong atendiendo a si el emparejamiento de las formas se hace utilizando características globales o locales.
Ilustración extraída del artículo [15].

2.2.1. Descriptores de formas basados en contornos

Los descriptores de contorno basan su procesamiento en la información del contorno. A continuación presentaremos algunos de los métodos que son aplicables dentro del contexto del problema.

2.2.1.1. Shape context Shape context [17]es un descriptor de formas invariante a escala y rotación. Es similar a los descriptores de puntos importantes como SIFT, pero les aventaja en que pueden describir objetos con poca información de texturas. Shape context primero hace un muestreo de los puntos del contorno. Por cada punto, construye un histograma en coordenadas polares de los otros puntos del contorno, ubicando el centro del histograma en las coordenadas del punto que se está procesando. El histograma se suele dividir en doce regiones angulares (cubriendo así 3 regiones por cada cuadrante) y varias regiones longitudinales. De esta manera las coordenadas de los otros puntos caen en una de esta regiones creando una firma característica; ya que el contexto de cada punto se computa contabilizando el número de puntos de cada región. Finalmente el proceso de emparejamiento se basa en encontrar para cada punto de la primera forma, un único punto de la segunda forma cuyo contexto de forma sea lo más similar posible.

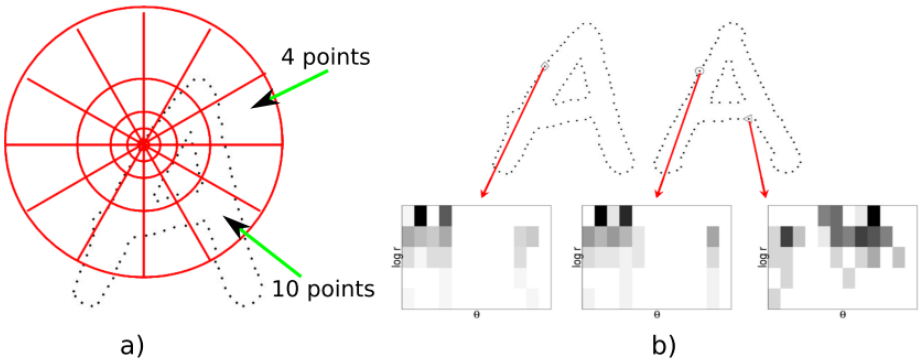


Fig. 18: (a) Ejemplo del contexto de forma de un punto (b) Contextos de forma de 3 puntos.
Notar la similitud del contexto entre los dos primeros puntos. Ilustración extraída del artículo [15].

2.2.1.2. Modelo de contornos fragmentados o "Boundary Fragment Model" El modelo de contornos fragmentados[18] es un descriptor invariante a traslación y parcialmente a rotación y su funcionamiento consiste en dos etapas. En la primera toma como fuente los contornos fragmentados obtenidos de la aplicación de un detector de bordes. A partir de aquí, detecta fragmentos de contornos disjuntos a través de un proceso de clustering. De esta manera crea un libro de códigos o "codebook" con las propiedades de

cada fragmento del contorno y a veces también con las relaciones espaciales entre los fragmentos. En la segunda etapa, la de detección, inicialmente se buscan todos los fragmentos de la imagen procesada que están contenidos en el libro de códigos. Para posteriormente ponderar los fragmentos modelos y detectar si están presentes en la imagen procesada.

Hay varias implementaciones de los contornos fragmentados, diferenciándose entre sí según la codificación del libro. Y más concretamente , se diferencian en lo estricto que es el modelo que representación la figura. Por ejemplo hay versiones que utilizan un conjunto de palabras "bag of words" compuesto por descriptores de puntos de interés y otros más complejos que modelizan la figura con modelos gausianos.

2.2.1.3. k-Segmento Adyacente o "k-Adjacent Segment" (kAS)

kAS es un método elaborado por Ferrari et al. [19] que soluciona uno de los inconvenientes de la aproximación de los contornos fragmentados, y este es la falta de reutilización de los contornos fragmentados al ser muy específico a las formas que procesan. En cambio, no es invariante a rotación. Primero se agrupan los segmentos en grupos de k segmentos conectados en la red de segmentos de cotornos o "Contour Segment Network" (red desarrollada previamente por Ferrari [20]) que codifica la relación de conectividad entre los kAS grupos. Al ser variable el valor de k, es decir, el tamaño de los grupos de segmentos conectados, se pueden localizar uniones de contornos particulares. Por ejemplo con un valor de k = 3 se podría detectar igualmente un triangulo o un contorno con forma de Z. Además se requiere que sea entrenado con imágenes etiquetadas con un rectángulo delimitando la forma a localizar.

2.2.1.4. Espacio de escala de curvatura o "Curvature scale space" (CSS)

CSS [21] se basa en calcular los puntos máximos de inflexión de la curvatura del contorno y utilizarlos para describir la forma. Para calcular la curvatura aplican un kernel gausiano a cada punto de la curvatura mediante una operación de convolución incrementando el tamaño del kernel cada iteración hasta que no se hallen puntos de inflexión. Una vez llegado a este punto, se crea la descripción de la forma robusta a los cambios de escala y rotación. Esta descripción equivale a representar los puntos de inflexión en una gráfica considerando la posición de cada uno en la curva como eje de abscisas y el valor del tamaño del kernel como eje de coordenadas. En la siguiente imagen muestro los descriptores de tres versiones modificadas de la figura de un tiburón.

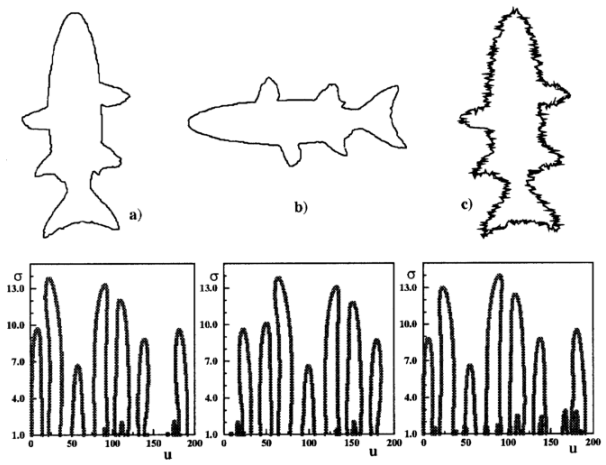


Fig. 19: Ejemplo de descriptores CSS
(a) Versión original (b) forma rotada (c) forma original con distorsión. Ilustración extraída de la página web [22].

2.2.1.5. Turning function A pesar de que no aparece en la clasificación hecha por Anh Phuong lo comento porque lo utilizamos en los trabajos previos [9] obteniendo muy buenos resultados. "Turning function" [23]es un método que describe la similitud entre dos polígonos proporcionando, por un lado, la distancia de similitud entre los polígonos ,y por otro, el ángulo relativo entre ellos. Primero se crea un modelo de la curvatura de cada polígono y posteriormente se procesa cada punto de la curvatura calculando el ángulo de la recta tangente con respecto a una dirección de referencia, que suele ser el eje x. De esta manera se ha creado la turning function que representa la variación del ángulo a lo largo de los puntos de curvatura del polígono; además de mostrar claramente donde se sitúan los vértices de la figura al apreciarse una diferencia entre los ángulos de lados adyacentes a un vértice cualquiera. Además hay que indicar que las turning functions son idénticas a cambios de escala y rotación.

La distancia de similitud entre las figuras se calcula minimizando la función de distancia entre las dos turning functions. Muy a groso modo, el cálculo se puede asemejar a superponer la primera turning function sobre la segunda adaptando, para cada tramo, el ancho y la altura. Es decir, en cada tramo, a la primera turning function se le aplica dos traslaciones (una por cada eje) para ajustarla lo mejor a la segunda turning. La traslación en el eje X efectuado sobre la primera turning function indica el cambio de longitud de esta para adapartase a la segunda; y del mismo modo la traslación en el eje indica el cambio de rotación. Muestro el proceso en la siguiente imagen.

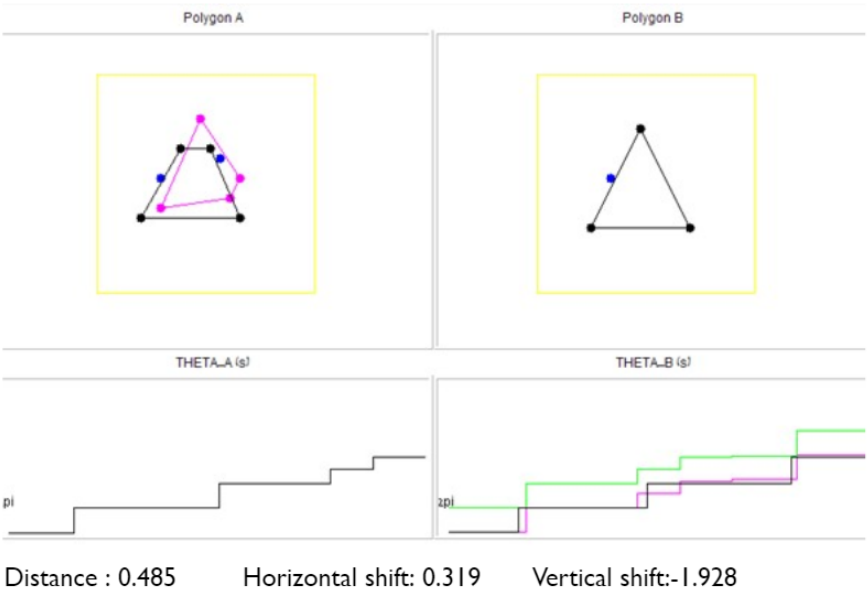


Fig. 20: Cálculo de distancia de similitud para dos turning function.
En violeta se representan las traslaciones efectuadas en el eje X sobre la turning function de la izquierda. Y en verde las traslaciones sobre el eje Y. Ilustración extraída de la página web [24].

2.2.2. Descriptores de formas basados en regiones

En [8] describen que los métodos basados regiones se basan en la forma de cada región con mínimo significado, es decir, utilizan la información codificada dentro de las regiones del objeto. Normalmente, estas técnicas se basan en cualquier tipo de blob o región dentro del conjunto de las regiones halladas en la etapa de segmentación de imágenes. En cambio no utilizan características de bajo nivel como texturas y colores. En general son menos sensibles al ruido en comparación con los descriptores de contorno, pero es necesario el uso que los algoritmos de segmentación sean fiables.

Un método bien conocido, es la aplicación de la transformada de Hough (TH) en cualquiera de sus versiones:

- TH Clásica: utilizada para detectar segmentos rectos.
- TH Generalizada (GHT): utilizada para detectar una forma específica (círculos, arcos, etc)
- TH Probabilística: optimización de la versión clásica.

Por ejemplo he observado en [7] una aplicación interesante para abordar nuestro problema. En ella utilizan una modificación de GHT para la extracción de grandes conjuntos de datos en petroglifos. Comentan que tienen buenos resultados con petroglifos que están formados por varias partes que suelen ser distintas entre sí.

2.2.3. Descriptores de formas basados en puntos

Los métodos basados en puntos importantes o "Key Points" detectan zonas concretas que represente un punto clave de la imagen. Se define como punto importante aquel que es relevante por la cantidad de información de su entorno, por ejemplo por la existencia de bordes o texturas. Pero también aquel punto que es invariante a cambios de escala o rotación u otro tipo de perturbación local o global. SIFT [25] ("Scale-Invariant Feature Transform") es uno de los primeros descriptores exitosos capaz de encontrar características invariante partiendo de los vértices que le proporciona un algoritmo de detección de esquinas como el de Harris [26]. SIFT funciona bastante bien y es muy preciso, pero su ejecución requiere un alto coste computacional. Versiones similares ha surgido en los últimos años mejorando la eficiencia computacional de SIFT. Entre ellos podemos enumerar a SURB [27] ("Speeded Up Robust Features"), ORB [28] ("Oriented FAST and Rotated BRIEF") o HOG [29] ("Histogram of Oriented Gradients"). En general estos descriptores consiguen buenos resultados reconociendo objetos, pero trabajan con texturas por lo que no sirven dentro del contexto de este trabajo. Seguidamente muestro un ejemplo del emparejamiento de puntos, entre una imagen y ella misma rotada, utilizando el descriptor ORB.

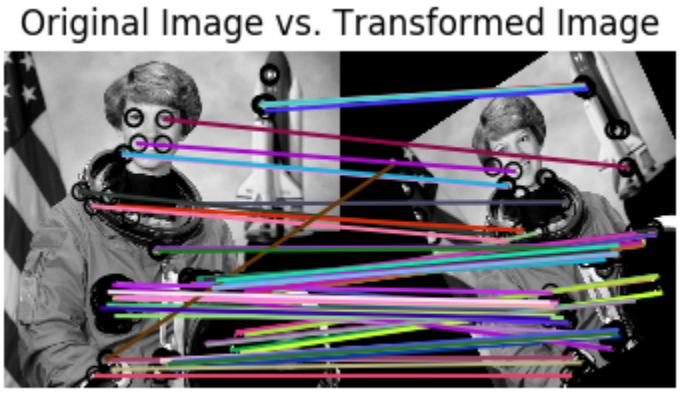


Fig. 21: [30].

Otro método popular basado en puntos es el descriptor de filtros de Gabor [31]. Ha sido utilizado con éxito en el emparejamiento de imágenes en aplicaciones como el reconocimiento óptico de caracteres, reconocimiento del iris o de huellas dactilares. Funcionan muy bien con imágenes donde la orientación de los píxeles es muy importante. Por ejemplo también se utilizan en la compresión de imágenes o en la segmentación de texturas. Otra técnica es RANSAC "RANdom SAmple Consensus", que es un método iterativo para estimar los parámetros de un modelo matemático. Son muy utilizados en visión artificial principalmente cuando hay que procesar dos imágenes muy similares en donde hay una transformación geométrica entre las dos. Por ejemplo se puede aplicar como alternativa a los descriptores basados en en puntos de interés.

2.3. Uso de técnicas de graph matching en problemas de reconocimientos de patrones en imágenes

Los autores del artículo [15] terminan aplicando técnicas de graph matching a su problema de reconocimiento de objetos en sketch. Exponen que, a pesar de las distintas metas de las aplicaciones de reconocimiento de patrones, en última instancia las aplicaciones tienen que resolver problemas que comparten varias propiedades como por ejemplo el tener que lidiar con transformaciones no rígidas entre el patrón modelo y el analizado; o la necesidad de medir la similaridad entre los objetos. Y afirman que una de las ventajas de utilizar graph matching es el fácil manejo de la transformaciones no rígidas, frente a otras técnicas comentadas con anterioridad, como por ejemplo con el método RANSAC.

En una gran variedad de trabajos previos sobre reconocimiento de patrones vemos que el uso de grafos relacionales está bastante extendido. Por ejemplo en los trabajos de Kopczynski y Sester [14] vuelven a utilizar un grafo relacional para representar la estructura de un sketch. Bunke [33] comenta que en las aplicaciones de reconocimiento de patrones y visión artificial es requerido medir la similaridad entre los objetos a reconocer y que los grafos son una buena herramienta para realizar esto. Y por consiguiente, tendremos que resolver el problema de similaridad de grafos derivado de los problemas que teníamos con la medida de la similaridad en los objetos que han sido representados como grafos con anterioridad. Expone que las primeras aproximaciones de graph matching estaban restringidas a encontrar grafos o subgrafos isomorfos entre dos grafos [34]. Y que no tenían en cuenta el ruido o las distorsiones que se puedan encontrar en los objetos originales de los que se crea el grafo. Por lo que su aplicación estaba bastante restringida a casos en los que no hubiese una distorsión original. Por lo tanto se hace necesario tener en cuenta cierto grado de tolerancia al error durante el proceso de emparejamiento.

Brunke comenta que se han desarrollado una gran cantidad de algoritmos capaces de resolver el problema del emparejamiento de grafos a partir del algoritmo original de Ullman [34]. La mayoría de estos algoritmos se basan en la búsqueda heurística dentro del árbol de búsqueda. Incorporan varias técnicas heurísticas de vuelta atrás o Backtraking para ir podando consecutivamente el espacio de búsqueda. Estos algoritmos garantizan encontrar una solución óptima en el emparejamiento pero a cambio requieren un tiempo y espacio exponencial debido a la complejidad NP-Completo del problema. Para reducir el tiempo y espacio de computación se han desarrollado varios algoritmos que proponen una solución aproximada. Existe multitud de algoritmos aproximativos basados en distintas técnicas de Inteligencia Artificial. Entre ellos encontramos redes neuronales, algoritmos genéticos, métodos aproximados basados en el flujo máximo, redes bayesianas, etc. Además todos estos algoritmos, cuando se ejecutan, podrían retornar una solución que fuese un "mínimo local", es decir, pueden quedarse atascados en un mínimo local y así no devolver una solución óptima. Para solucionarlo Brunke expone una serie de técnicas de emparejamiento de árboles que proponen correcciones del error al inicio de la creación del árbol decisión o durante su ejecución.

Bunke concluye que si queremos adaptar un algoritmo de emparejamientos de grafos a una tarea particular debemos resolver dos problemas concretos. El primero de ellos es el desarrollo de un grafo válido que represente los objetos del dominio del problema que tienen que ser encontrados. Y segundo que hay que definir uno operación de corrección del error unida al coste que esta corrección tiene.

En [35] y [36] encontramos otra técnica para el emparejamiento de grafos. En este caso se propone considerar el emparejamiento de grafos como un problema de optimización combinacional con restricciones. Se propone el uso de algoritmos de distribución estimada ("Estimation Distribution Algorithms, EDAs). Obteniéndose mejores resultados que con las aproximaciones tradicionales como los algoritmos genéticos. Esta propuesta se engloba dentro de las últimas propuestas descritas por Bunke, en donde utilizamos una operación de coste del error combinada con cualquier tipo de algoritmo. Esto también lo vemos reflejado en [36] ya que expone que por primera vez se han realizado adaptaciones del esquema de muestreo lógico probabilístico o "probabilistic logic sampling (PLS)" sobre la red bayesiana. Permitiendo así que se pueden aplicar restricciones en el algoritmo EDAs original.

3. Descripción de la solución

En esta sección describo el método desarrollado. He decidido descomponer la descripción en las etapas necesarias para resolver el problema en cuestión. Básicamente son una extensión de las etapas de los trabajos previos descritas en la sección 1.4. La etapas de mi solución son:

1. Preprocesamiento de las imágenes: se realiza el procesamiento de todas las imágenes originales para crear una imagen normalizada de la que extraer el grafo.
2. Creación de los grafos:
   a) Aproximación de los contornos de una figura.
      1) Utilizando las esquinas de Harris.
      2) Usando el algoritmo de Ramer-Douglas-Peucker.
   b) Simplificación de vértices y creación del grafo base.
   c) Cálculo de los atributos de los nodos.
3. Cálculo de la similitud entre el patrón y el dibujo:
   a) Cálculo de las matrices de coste.
   b) Métricas utilizadas en el diagnóstico del DCL .

3.1. Preprocesamiento de las imágenes

3.1.1. Planteamiento

La etapa de preprocesamiento es importante porque se crean las imágenes de partida para el resto de las etapas. Es importante preprocesarlas por varios motivos. Por una parte no todas son óptimas para ser segmentadas porque hay imágenes donde es difícil apreciar el contorno a simple vista (provocado por defectos de la digitalización). En la siguiente figura muestro varios dibujos con trazos débiles.

Fig. 22: Ejemplos de dibujos originales con trazos muy débiles.

Otro motivo es la diferencia de tamaños en una misma clase. Lo cuál añadiría mayor complejidad al proceso de Graph Matching al compararse imágenes similares en forma pero con tamaños distintos. Además también hay cierta diferencia de tamaño entre los dibujos modelos presentados a los pacientes, es decir, no todos los pacientes han realizado la copia del modelo bajo las mismas referencias. Pero también es verdad que la normalización de tamaños hace que se pierda la información del tamaño original y por tanto su significación como posible métrica discriminatoria. En la siguiente figura vemos un ejemplo de las diferencias de tamaño entre dos sesiones realizadas por un mismo paciente.

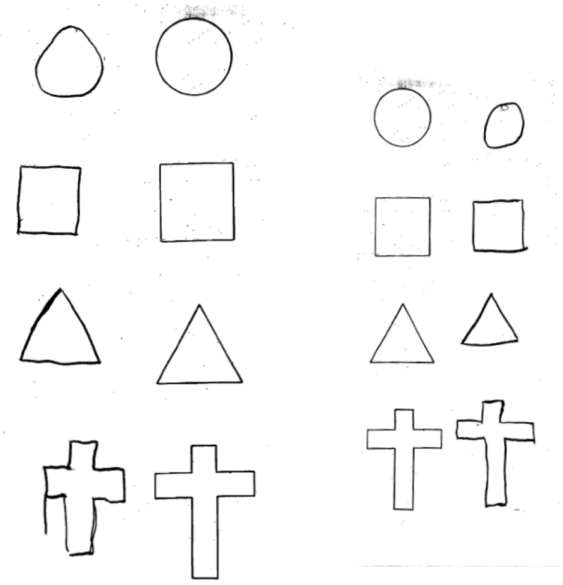


Fig. 23: Ejemplo de las diferencias de tamaño entre dos sesiones realizadas por un mismo paciente.

La normalización de tamaños va a venir marcada por el tamaño de los dibujos modelos. Es decir, se ha seleccionado un dibujo modelo por cada clase y es este tamaño el cuál van a tener todos los dibujos de los pacientes. Seguidamente expongo como he desarrollado el preprocesamiento.

3.1.2. Implementación

Las técnicas utilizadas en esta etapa son las habituales del preprocesamiento de imágenes. Se comienza con un filtro gaussiano para eliminar el ruido de este tipo. Continua creándose las imágenes binarias utilizando la umbralización de Otsu [41]. En este punto es importante destacar que se utiliza Otsu dos veces sobre la imagen original. Primero con una umbralización en dos niveles y otra con tres, es decir, se divide el histograma de intensidades en dos y tres niveles. A su vez, la de tres se transforma en dos niveles etiquetando los píxeles de nivel tres como dos. Después de esta transformación se selecciona la

umbralización más adecuada. Por defecto se selecciona la de tres si su tamaño es similar al de dos. Esto es porque la umbralización de tres niveles recoge mejor los trazos débiles pero en otras ocasiones aparece más ruido (proveniente de la digitalización). En la sección de 4.1 mostraré ejemplos de las dos umbralizaciones.

Una vez efectuada la umbralización ya se dispone de una imagen binaria adecuada. El proceso continúa eliminando las regiones cuyo área es inferior a un umbral con valor de 40 píxeles establecido mediante ensayo y error. De esta manera se consigue eliminar la mayor parte del ruido proveniente del proceso de digitalización (generalmente como puntos aislados). La imagen filtrada puede seguir conteniendo ruido proveniente de la digitalización y de la umbralización adaptativa pero ya será el proceso de Graph Matching el encargado de lidiar con este tipo de ruido.

Ahora partiendo de la imagen filtrada se realiza un recorte de esta **(1)**. De esta manera tenemos una imagen adecuada para ser escalada al tamaño de la imagen modelo. La idea es que se mantenga la relación de aspecto en el escalado para que no se provoquen deformaciones. Aunque previamente a escalar la imagen al tamaño correspondiente del modelo, se realiza una operación de traslación sobre **(1)** creándose una imagen cuadrada **(2)** cuyo lado vale la hipotenusa de la imagen **(1)** . Es decir, el valor del lado de **(2)** equivale a $h = \sqrt{a^2+b^2}$ (a y b son las dimensiones de la imagen **(1)** en el eje horizontal **x** y vertical **y** respectivamente); y la operación de traslación se define como la transformación de los píxeles de **(1)** con coordenadas $p(x_1, y_{1)}$ a las nuevas coordenadas de **(2)** $p(x_2, y_{2)}$ según $p(x_2, y_2) = p(x_1 + \frac{h-a}{2}, y_1 + \frac{h-b}{2})$.

Los pasos anteriores se aplican a todas las imágenes modelos. En el caso de una imagen modelo se guarda la imagen **(2)** y su tamaño. Si es una de un paciente se hace un escalado con una interpolación cúbica y se obtiene la imagen **(3)**, la cuál ya estaría normalizada. A continuación muestro el proceso resumido en la siguiente figura.

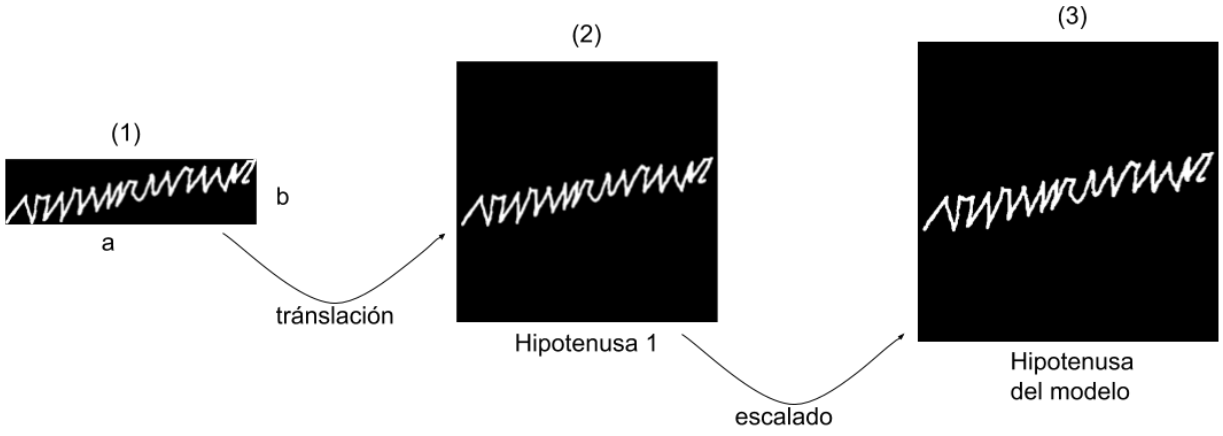


Fig. 24: Proceso de traslación y escalado de una imagen del paciente

3.2. Creación de los grafos

3.2.1. Planteamiento

Esta es una de las etapas más importantes de la solución debido a que la manera en la que se cree el grafo afectará en gran medida al resto de etapas. Por ejemplo si creamos un grafo muy preciso pero con gran cantidad de nodos y enlaces, es muy posible que el cálculo de la similitud sea inviable por el alto coste computacional. Otro caso puede darse cuando el grafo es lo suficientemente preciso en varias zonas pero muy impreciso en otras. En este caso puede que se de un valor de similitud adecuado, pero a la vez, se ha prescindido de la información de las zonas imprecisas. Por tanto, las métricas discriminatorias entre tipos de pacientes no serán del todo verosímiles.

Y en esencia la creación del grafo depende de como se implemente las extracción de la información. En este trabajo se van a extraer los segmentos rectos de cada figura. Esto es apropiado para la mayoría de las figuras por que solo contienen formas rectas. En cambio esto plantea dificultades con los círculos y los bucles, pero considero que es suficiente una aproximación basada en segmentos rectilíneos. Todos los trazos de cada figura están contenidos únicamente en los contornos (exteriores e interiores) porque ninguna figura esta rellena. Por lo que los contornos representan la información primordial para la creación de los grafos. Básicamente el método aproxima cada contorno en segmentos rectos y contiguos y a partir de ellos se crea el grafo tomando los vértices de los segmentos como nodos y los propios segmentos como enlaces.

3.2.2. Teoría

En este apartado voy a exponer la definición formal de un grafo. Expongo la definición de [37] porque es la nomenclatura que utilizo con los grafos a lo largo de este trabajo.

Definición. Un grafo g es una 4-tupla (lista ordenada de 4 elementos) de $g = \{V, E, \mu, v\}$donde:

- V es el conjunto finito de nodos,
- $E \subseteq V \times V$ es el conjunto de enlaces,
- $\mu : V \to L$ es la función de etiquetación de los nodos, y
- $v : E \to L$ es la función de etiquetación de los enlaces.

Esta función nos permite utilizar grafos arbitrarios sin restricciones en las funciones de etiquetación. Por ejemplo, podemos definir las etiquetas como un conjunto de enteros (R^n), o como un conjunto de etiquetas simbólicas $L = \{\alpha, \beta, \gamma, \ldots\}$. Además, un grafo

sin etiquetas es obtenido como un caso especial asignando la misma etiqueta l a todos los nodos. Los enlaces están formados por pares de nodos (u, v), donde $u \in V$ es el nodo origen y $v \in E$ el nodo destino. Los grafos no dirigidos son modelados insertando un enlace inverso $(v, u) \in E$ por cada enlace $(u, v) \in E$ teniendo $v(u, v) = v(v, u)$.

En este trabajo vamos a trabajar con grafos no dirigidos y con atributos en los nodos y enlaces. A continuación expongo la implementación que he seguido para generar los grafos.

3.2.3. Implementación

Para abordar la generación del grafo he dividido en proceso en varías etapas (el procedimiento es idéntico para los dibujos modelos y para los dibujos de los pacientes):

1. Aproximación de los contornos de una figura
2. Simplificación de vértices y creación del grafo base
3. Cálculo de los atributos de los nodos

3.2.3.1. Aproximación de los contornos de una figura Para extraer los segmentos rectos primero se comienza realizando la esqueletización de la imágenes procesadas para reducir el tamaño de los trazos a un pixel de ancho sin perder gran parte de la información geométrica. Aunque la esqueletización es muy sensible al ruido de contorno y puede originar trazos pequeños que aparecen como ramas del trazo principal. Esto depende del algoritmo de esqueletización, es este caso se ha elegido el algoritmo de Lee [44] al generar menos ruido.

Con la imagen preprocesada ya esqueletizada pasamos a aproximar todos los contornos que definen la imagen. Se emplea un algoritmo recursivo que en cada iteración extrae el contorno exterior. Una vez almacenado se elimina y se vuelve a llamar al algoritmo hasta que ya no quedan contornos. La ventaja de extraerlos recursivamente radica (sobre todo en las imágenes con contornos interiores) en que cada contorno se aproxima una vez en comparación con extraer todos los contornos a la vez. Por lo que se simplifican las duplicidades de contornos.

Ahora el proceso continua con la aproximación de los contornos extraídos. Hay dos procedimientos dependiendo del tipo de figura. Para todas las figuras a excepción de los círculos, se calculan las esquinas más relevantes siguiendo el algoritmo de Harris [26] y se toman como base para la aproximación de los contornos. En el caso los círculos se utiliza el algoritmo de Ramer-Douglas-Peucker (RDP) [42], [43] . A continuación expongo los dos procedimientos.

Aproximación de contornos utilizando el algoritmo RDP Esta aproximación realiza un proceso iterativo por cada contorno exterior detectado. Cada contorno se aproxima utilizando en algoritmo de RDP. Para cada contorno simplificado se guardan en una lista los segmentos aproximados por el algoritmo.

Aproximación de contornos con las esquinas de Harris El procedimiento descrito a continuación se ha elegido porque permite representar adecuadamente los trazos ligeramente curvados o no perfectamente rectos propios de la realización de un dibujo a mano alzada. Ya que los puntos significativos dónde se producen estos defectos son detectados mejor por el algoritmo de esquinas que con RDP. Pero también existe la problemática de que no se detecten suficientes esquinas por lo que la aproximación sería más inexacta. Este es el caso de los círculos, al detectarse pocas esquinas se producen varias deformaciones.

Al igual que en la aproximación anterior, se aproxima iterativamente cada contorno exterior detectado. A continuación se calculan las coordenadas de las esquinas con el algoritmo de Harris. A partir de las las esquinas se buscan los puntos del contorno cercanos a cada esquina. Para de este modo poder detectar los intervalos del contorno que se hayan entre dos esquinas y así simplificar dicho intervalo por un segmento formado por las respectivas esquinas. En el caso de que haya puntos del contorno que no se encuentren cerca de una esquina, el procedimiento aproxima estas regiones del contorno con el algoritmo RDP. También hay que tener en cuenta que en algunas situaciones no es válido el segmento de esquinas que aproxima un intervalo de puntos del contorno. Esto se debe a que los puntos de intervalo del contorno forman un trazo curvo que no es colineal con el segmento de las dos esquinas. Por tanto es necesario comprobar que los segmento de esquinas son colineales con el intervalo que aproximan (función de **segmento_correcto**). A continuación se expone el pseudocódigo del algoritmo implementado.

Algoritmo 1 Algoritmo de aproximación de contornos utilizando las esquinas de Harris.

1: **INPUTS:**
2: *contorno:* Lista de contornos exteriores.
3: *coords_Hs:* Lista de las coordenadas de las esquinas de Harris.
4: *umbral_distancia:* Umbral para considerar válida la distancia cercana de una esquina de Harris a un punto del contorno.
5: *umbral_semento_OK:* Umbral para considerar válido un segmento aproximado entre dos esquinas.
6: **OUPUTS:**
7: *segmentos:* Umbral para considerar válido un segmento aproximado entre dos esquinas.

8: **for** each: *contorno* de *contornos* **do**
9: **for** each: *coord* de *contorno*, *coord_Hs* de *coords_Hs* **do**
10: **if** distancia euclidiana (*coord*, *coords_Hs*) $\leq$ *umbral_distancia* **then**

11: Añadimos a la *lista_A* las ternas válidas según (*idx_coord_cont*, *idx_esquina*, *distancia*).
12: **end if**
13: **end for**
14: **if** existe *lista_A* **then**
15: Creamos la *lista_B* con los intervalos de las ternas de la *lista_A* que tienen el mismo *idx_esquina*. Es decir, cada elemento de la *lista_B* representa: (posición inicial de la terna i, número de ternas con el mismo *idx_esquina*).
16: **for** each: intervalo de *lista B* **do**
17: Obtenemos de la *lista_A* las ternas de cada intervalo.
18: Obtenemos las ternas con menor distancia.
19: Las ordenamos en orden ascendente según el *índice_coord_cont*.
20: Añadimos en la *lista_D* la primera terna del conjunto ordenado con anterioridad.
21: **end for**
22: **if** el primer elemento de la *lista_D* contiene el *idx_coord_cont* igual a 0 **then** ▷ (índice del primer punto de *contorno*)
23: **for** *i* desde 0 hasta la longitud de la *lista_D* **do**
24: Creamos *segmento_Hs* formado por las dos *coords_Hs* con los índices *i* e *i + 1* sacados de *idx_esquina* de la *lista_D*. Es decir, se añaden las dos coordenadas según: *coords_Hs (idx_esquina (i)) y coords_Hs (idx_esquina (i + 1))*.
25: Creamos *ptos_cont_intervalo* con todas las coordenadas del contorno dentro del intervalo según: *contorno (idx_coord_cont (i))* hasta *contorno (idx_coord_cont (i + 1))*
26: SEGMENTO_CORRECTO(*segmento_Hs, ptos_cont_intervalo, segmentos, umbral_semento_OK*)
27: **end for**
28: **else**
29: Guardamos en *idx_cont_NO_Hs* los índices iniciales del contorno que no están en la *lista_D* (todos los índices del contorno hasta el primero de la *lista_D*).
30: Guardamos en *idx_cont_Hs* todos índices del contorno que están en la *lista_D*.
31: APROXIMACIÓN_RDP_NO_HARRIS(*idx_cont_NO_Hs*, *idx_cont_Hs*)
32: Aproximamos las coordenadas del contorno recogidas en la *lista_D* aplicando el procedimiento descrito en las líneas 22 a 26.
33: Guardamos en *idx_cont_NO_Hs* los índices finales del contorno que no están en la *lista_D* (todos los índices del contorno desde el siguiente al último de la *lista_D* hasta el último índice del contorno).
34: APROXIMACIÓN_RDP_NO_HARRIS(*idx_cont_NO_Hs*, *idx_cont_Hs*)
35: **end if**
36: **else**
37: En caso de que no exista la *lista_A* aproximamos todos los puntos del contorno con el algoritmo de RDP.
38: **end if**

end for

function SEGMENTO_CORRECTO(*segmento_Hs, ptos_cont_intervalo, segmentos, umbral_semento_OK*)
    Creamos la imagen *img_segmento_Hs* dibujando únicamente *segmento_Hs*.
    Creamos la imagen *img_cont* con todos los puntos del contorno del intervalo *ptos_cont_intervalo*.
    Creamos la imagen *img_AND* resultante de la operación AND entre *img_cont* y *img_segmento_Hs*
    Calculamos *tamaño_AND* y *tamaño_segmento_Hs* como el número de píxeles de *img_AND* y *img_segmento_Hs*.
    if *tamaño_AND* / *tamaño_segmento_Hs* $\leq$ *umbral_semento_OK* **then**
        Aproximamos *ptos_cont_intervalo* con RDP y añadimos los segmentos aproximados a *segmentos*
    elseAñadimos *segmento_Hs* a *segmentos*
    end if
    return *segmentos*
end function

function APROXIMACIÓN_RDP_NO_HARRIS(*idx_cont_NO_Hs*, *idx_cont_Hs*)
    if no hay ningún *idx_cont_NO* en *idx_cont_Hs* **then**
        Aproximamos las coordenadas del contorno recogidas en *idx_cont_NO_Hs* con el algoritmo RDP y añadimos los segmentos aproximados a *segmentos*.
    end if
    return *segmentos*
end function

A continuación (figura 25) se muestran algunos ejemplos de las aproximaciones obtenidas. En la fila de arriba se representan las imágenes preprocesadas indicando las esquinas detectadas (puntos en rojo) y en la fila de abajo los segmentos aproximados. Estos aparecen en varios colores para diferenciar unos segmentos de otros. En general las aproximaciones recogen adecuadamente los trazos de los dibujo, incluyendo las pequeñas deformaciones, aunque algunas formas originales se han perdido como es el caso del humo de la casa.

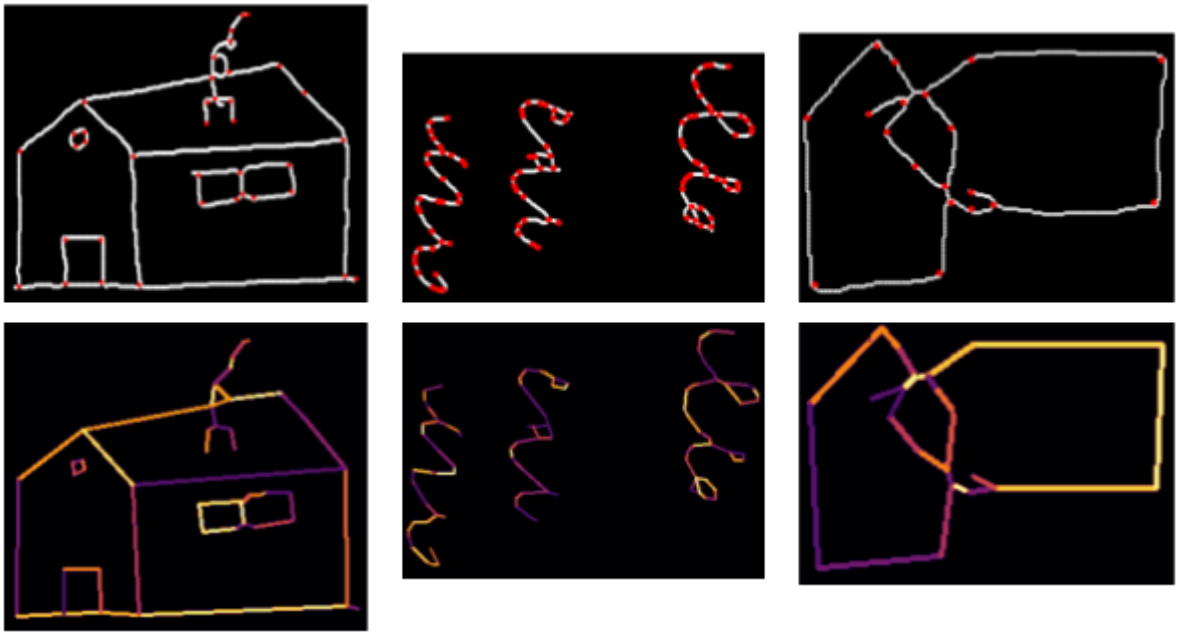


Fig. 25: Ejemplos de la simplificación de los contornos.

Y para concluir la sección de aproximación, en la figura 26 vemos un ejemplo de la simplificación de los círculos. A la izquierda se representa el círculo preprocesado juntos con las esquinas de Harris detectadas (puntos en rojo). En la imagen del centro y de la derecha están las simplificaciones por el método del algoritmo 1 y RDP respectivamente. Se observa como la simplificación del algoritmo 1 ha originado más distorsiones en comparación con la de RDP.

Fig. 26: Ejemplos de la simplificación de los contornos de los círculos.

3.2.3.2. Simplificación de vértices y creación del grafo base La creación del grafo es directa a partir de los segmentos aproximados. Los extremos de los segmentos suponen los nodos del grafo y los enlaces vienen determinados por qué segmentos son concurrentes en cada extremo de los segmentos. Aunque previo a este paso se realiza una simplificación

agrupando aquellos extremos que están muy cerca. Primero se crea una imagen con las coordenadas de todos los puntos extremos que definen los segmentos de la lista obtenida en el paso anterior. Después se realiza una dilatación de la imagen anterior para agrupar las distintas coordenadas que toma cada extremo. Se utiliza un "kernel" en forma de disco cuyo tamaño es definido en un parámetro de entrada, el cuál es calculado a través de ensayo y error (se establece un valor de 4). Este es uno de los parámetros de entrada más importantes porque su valor influye bastante en el grafo creado. Valores altos harán que se genere un grafo muy simplificado y a la inversa cuando los valores son pequeños.

Con todo esto ya tenemos la información necesaria para crear el grafo. Falta obtener la relación de correspondencia entre cada extremo de la imagen sin dilatar y su correspondiente agrupación en la imagen dilatada. Para ello, primero se etiqueta la imagen dilatada y se pasa a calcular los centros de masas de cada agrupación. Posteriormente, para cada segmento aproximado se hallan las etiquetas y los centros de masas correspondientes a las agrupaciones que se corresponden con cada punto extremo del segmento aproximado. Posteriormente se traslada esta información al grafo creando los nodos de los puntos extremos y un enlace con el atributo de longitud del segmento aproximado. Igualmente a cada nodo se le añade un atributo que representa las coordenadas del centro de masas que lo agrupa.

En la figura 27 vemos un ejemplo de las etapas de creación del grafo: **A** representa la imagen preprocesada y las esquinas detectadas, **B** los segmentos aproximados, **C** la imagen con los extremos de los segmentos aproximados, **D** la de las agrupaciones una vez dilatada **C**. Para este caso únicamente hay dos agrupaciones con más de un punto extremo (círculos en rojo en **D**) . El resto de puntos extremos de **C** no se han agrupado por estar separados. **E** representa la visualización del grafo base. En él cada nodo tiene como atributo las coordenadas de se posición y cada enlace la longitud del mismo. En el siguiente apartado se van a calcular más atributos a partir de esta información.

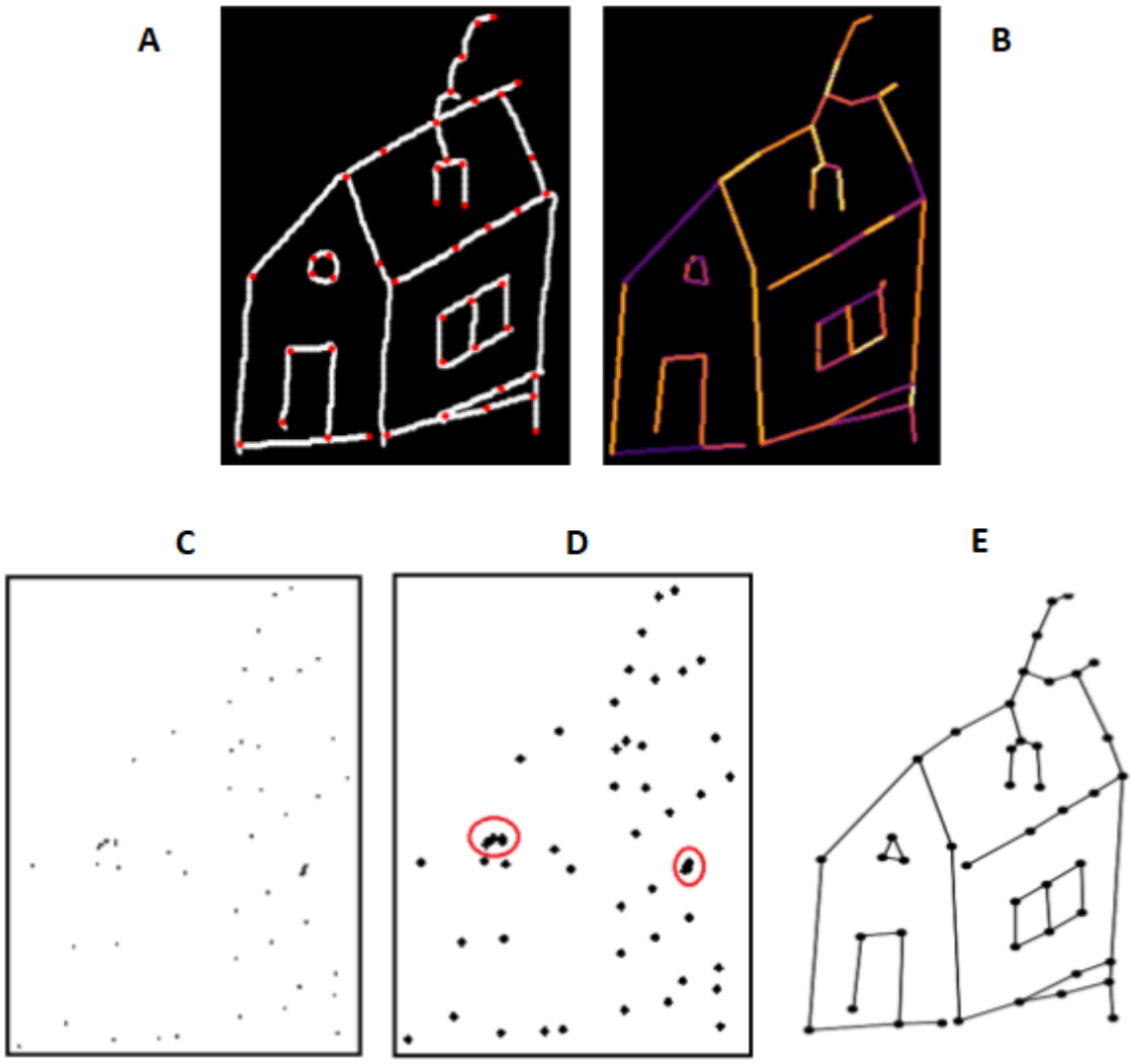


Fig. 27: Ejemplos de de las etapas de creación del grafo.

3.2.3.3. Cálculo de los atributos de los nodos Partiendo del grafo base creado en la etapa anterior, se calcula una serie atributos adicionales para ampliar la información del grafo. En definitiva cada grafo de un dibujo va a estar descrito por los siguientes atributos:

1. Atributos de los nodos:

   a) **Coordenadas del nodo**: representa las coordenadas del centro de masas de la agrupación a la que corresponde un nodo.

   b) **Betweennes Centrality**: representa la frecuencia de ocurrencia de dicho nodo entre los caminos más cortos "shortest path" que conectan pares de nodos y que

pasan por él. Nodos con un valor alto tendrán gran importancia en el grafo porque estos nodos están en el medio de los caminos que relacionan las áreas más importantes del grafo.

c) **Degree**: cuantifica el número de enlaces que llegan a un nodo

d) **Vector_CM**: representa el vector que une cada nodo con el centro de masas del dibujo. En la figura 28 se muestra los cuatro vectores del CM de masas de cada nodo para el modelo del cuadrado.

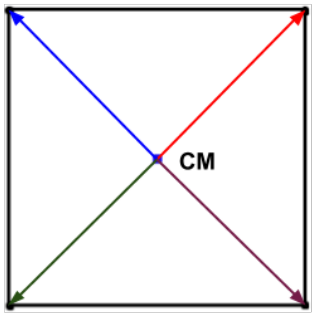


Fig. 28: Ejemplo del atributo Vector_CM para la imagen del cuadrado modelo.

e) **Vector_Suma**: viene a representar el vector resultante de sumar vectorialmente todos los vectores que parten de un nodo. Volvemos a utilizar la imagen del cuadrado, figura 29, modelo para representar el vector suma (en naranja) como la suma de $\overline{V_1}$ y $\overline{V_2}$ (representados en azul). Es decir, representa la suma vectorial: $\overline{V_{suma}}$=$\overline{V_1} + \overline{V_2}$

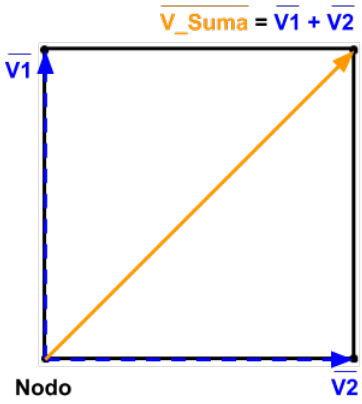


Fig. 29: Ejemplo del atributo Vector_Suma para un nodo de la imagen del cuadrado modelo.

f) **Ángulos_nodo**: está representado por una lista que contiene el valor de todos los ángulos de los enlaces que llegan a un nodo tomando como referencia el eje de coordenadas sobre el nodo. Por ejemplo para el cubo modelo de la figura 30

el valor del atributo Ángulos_nodo del nodo interno vale {45º, 180º, 270º}. Es importante resaltar como las medidas de los ángulos se toman desde el eje de coordenadas X, Y que tiene como origen el nodo en cuestión. En caso de que un nodo no tenga enlaces Ángulos_nodo estará vacía.

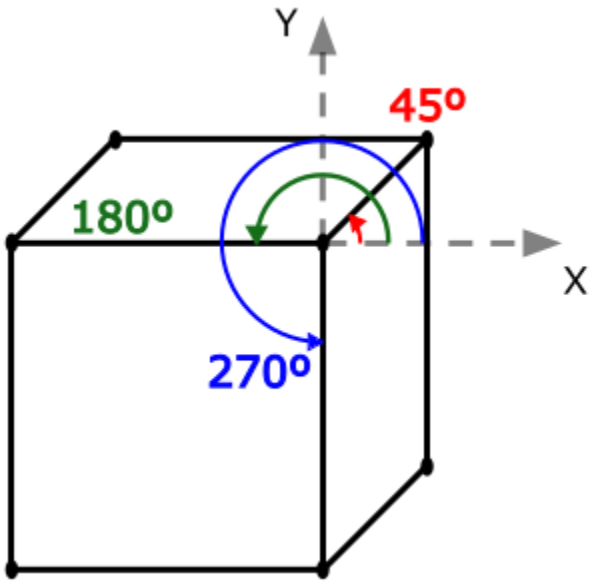


Fig. 30: Ejemplo del atributo Ángulos_nodo para un nodo de la imagen del cuadrado modelo.

2. Atributo de los enlaces:

   a) **Longitud**

3.3. Cálculo de la similitud entre el modelo y el dibujo

3.3.1. Planteamiento

En este apartado voy a exponer en detalle la teoría y la implementación desarrollada para calcular la semejanza entre el grafo modelo (g1) y el grafo del dibujo del paciente (g2) a través de técnicas de Graph Matching. Es una parte fundamental del método desarrollado en este trabajo porque a partir del cálculo de semejanza obtengo las métricas para el diagnóstico del DCL. Seguidamente expongo la teoría de Graph Matching. Me he basado en el artículo[37] porque es la base del método de Graph Matching que utilizo en este proyecto.

3.3.2. Teoría - Similitud de grafos

Cuando hablamos de graph matching nos estamos refiriendo al proceso de evaluar la similaridad estructural entre dos grafos. Existe dos tipos de graph matching atendiendo a la exactitud del emparejamiento. Si tenemos un emparejamiento exacto en donde todos los nodos han sido emparejados hablamos de "Exact Graph Matching". Y de "Inexact Graph Matching" cuando hay nodos en uno o ambos grafos sin emparejar.

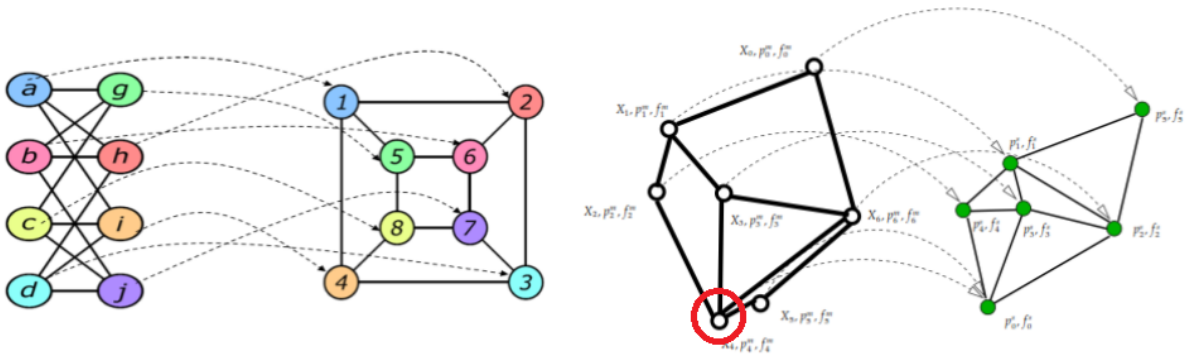


Fig. 31: Ejemplos de exact graph Matching (izquierda) y de inexact graph matching (derecha).
Podemos apreciar como en el Exact Graph Matching todos los nodos han sido emparejados y falta uno (círculo rojo) en el de Inexact Graph Matching.

En el contexto del problema se hace indispensable el unos de técnicas de las técnicas inexactas debido a la gran diferencias entre los dibujos. La distancia de edición de grafos o "Graph Edit Distance" (GED) es una de las técnicas inexactas más utilizadas debido a su flexibilidad y ha sido utilizada en una gran variedad de aplicaciones. GED nos proporciona un conjunto de operaciones de edición (conjunto de rutas de edición) llevadas a cabo para transformar g_1 en g_2. Por ejemplo si tenemos los dos grafos $\{g_1, g_2\}$de la siguiente

figura, un ejemplo de esta ruta, paso por paso y efectuando todas las ediciones sobre g_1: primero eliminar tres enlaces; luego eliminar un nodo; posteriormente insertar un nodo y seguidamente dos enlaces; para terminar sustituyendo dos nodos.

Fig. 32: Ejemplo del camino de ediciones que transforma el grafo g_1 en el grafo g_2. Ilustración sacada de [37]

Formalmente GED se define como:

Definición. (Distancia de edición de grafos - GED) Sea $g_1 = \{V_1, E_1, \mu_1, v_1\}$el grafo origen y $g_2 = \{V_2, E_2, \mu_2, v_2\}$el grafo destino. La distancia de edición de grafos entre g_1y g_2 se define como:

$$d(g_1, g_2) = \min_{(e_1,\dots,e_k) \in \gamma(g_1,g_2)} \sum_{i=1}^{k} c(e_i) \ ,$$

donde $\gamma(g_1, g_2)$ indica el conjunto de rutas de edición que transforman $g_1$en g_2 , y c indica la función de coste que mide la fuerza $c(e_i)$ de la operación de edición e_i.

A continuación muestro el algoritmo propuesto en [39] para el cálculo óptimo de GED. Concretamente se trata del algoritmo A^* que ha sido muy utilizado para resolver la distancia de edición de grafos. El cálculo se lleva a cabo mediante un algoritmo de búsqueda de árboles (primero-mejor) que explora el espacio de todos los posibles emparejamientos de los nodos y enlaces de g_1 con los de g_2. La idea básica es organizar el espacio de búsqueda como un árbol ordenado en donde el nodo raíz del árbol representa el punto de partida, los nodos interiores del árbol se corresponden con soluciones parciales, y los nodos de las hojas representan soluciones completas, que no son necesariamente óptimas. El árbol de búsqueda se construye mediante la creación iterativa de los nodos sucesores que son adyacentes al nodo que se está considerado en el árbol de búsqueda. Una vez que se ha hecho esta expansión se determina el nodo sucesor, es decir, el nodo por el que se continua expandiendo el árbol en la siguiente iteración, mediante una función heurística. Para un nodo$p$ en el árbol de búsqueda, se utiliza $g(p)$ para indicar el coste de la ruta de edición acumulada hasta ahora, y $h(p)$ para expresar el coste estimado desde p hasta un nodo de hoja. La suma $g(p) + h(p)$ proporciona el coste total asignado a un nodo abierto en el árbol de búsqueda. El algoritmo termina su búsqueda cuando llega a un nodo hoja cuya ruta de edición es completa, es decir, la ruta contiene todas las operaciones de edición necesarias para transformar g_1 en g_2. A^* garantiza que este coste final sea siempre optimo porque en cada posible exploración entre nodos del árbol, el algoritmo se queda con la ruta de edición que tiene menor coste. Esto lo vemos en la línea 8 del algoritmo en donde se elige la edición parcial $p \in OPEN$ que minimiza $g(p) + h(p)$.

Algoritmo 2 Algoritmo de distancia de edición de grafos A^* mostrado en [39]

INPUTS: Grafos no vacíos $g_1 = \{V_1, E_1, \mu_1, v_1\}$ y $g_2 = \{V_2, E_2, \mu_2, v_2\}$ donde $V_1 = \{u_1, \ldots, u_{|v_1|}\}$ y $V_2 = \{u_2, \ldots, u_{|v_2|}\}$
OUPUTS: Ruta de edición de coste mínimo de g_1 a g_2 p. ej. $p_{min} = \{u_1 \to v_7, u_2 \to \varepsilon, \ldots, \varepsilon \to v_1\}$
-
Inicializa OPEN como un conjunto vacío {}
Para cada nodo $w \in V_2$, inserta la sustitución $\{u_1 \to w\}$ en OPEN
Inserta la eliminación $\{u_1 \to \varepsilon\}$ en OPEN
loop
    Elimina $p_{min} = argmin_{p \in OPEN} \{g(p) + h(p)\}$ de OPEN
    if p_{min} es una ruta de edición completa **then**
        return p_{min} como la solución
    else
        Sea $p_{min} = \{u_1 \to v_{i\,1}, \ldots, u_k \to v_{i\,k}\}$
        if $k < |V_1|$ **then**
            Para cada $w \in V_2 \setminus \{v_{i\,1}, \ldots, v_{i\,k}\}$, inserta $p_{min} \cup \{u_{k+1} \to w\}$ en OPEN
            Inserta $p_{min} \cup \{u_{k+1} \to \varepsilon\}$ en OPEN
        else
            Inserta $p_{min} \cup \bigcup_{w \in V_2 \setminus \{v_{i\,1}, \ldots, v_{i\,k}\}}$ en OPEN
        end if
    end if
end loop

Aunque A^* encuentra una ruta de edición óptica, la complejidad computacional es exponencial con respecto al número de nodos que contienen los grafos. Por lo que los tiempos de computación y la memoria utilizada pueden ser muy grandes. Con motivo de reducir la complejidad, Bunke et Al. en [37] y [38] han aproximado GED a un problema de asignación, cuya definición formal es:

Definición. (Problema de Asignación). Asumimos que tenemos dos conjuntos A y B con una matriz de coste **C**de tamaño $n \times n$ compuesta por números reales donde $|A| = |B| = n$. El elemento de la matriz $\mathbf{C}_{ij}$ se corresponden con el coste de asignar el elemento i de A con el elemento j de B. El problema de asignación puede enunciarse como encontrar una permutación $p = p_1, \ldots, p_n$ de enteros $1, 2, \ldots, n$ que minimiza $\sum_{i=1}^{n} \mathbf{C}_{ip_i}$

Básicamente el problema consiste en encontrar la manera de asignar unos recursos a unas determinadas tareas. Se busca que esta asignación tenga el menor coste y teniendo en cuenta las restricciones de que solo se puede asignar un recurso a una única tarea y que cada tarea debe de ser ejecutada por un solo recurso. Existe un algoritmo, conocido como algoritmo húngaro, desarrollado por Harold W. Kuhn que resuelve el problema de asignación en un tiempo $O(n^3)$. Posteriormente James Munkres revisó la implementación

original y consiguió resolver la asignación en un tiempo fuertemente polinómico. El de Munkres es el utilizado en [37]. A continuación describo este algoritmo.

Algoritmo 3 Algoritmo de Munkres para el problema de asignación

Input
    Una matriz de coste C de dimensionalidad n
Output
    La asignación de nodos o enlaces con el mínimo coste
Para cada fila r en C , resta su elemento más pequeño a cada elemento de r
Para cada columna c en C , resta su elemento más pequeño a cada elemento de c
Para todos los ceros z_i en C , marca z_i con un asterisco (0^*) si no hay un cero con asterisco en su fila o columna
STEP 1:
for Cada columna conteniendo un 0^* **do**
    Cubre esta columna (marcar todas las celdas de la columna)
end for
if n columnas estas cubiertas **then**
    GOTO DONE
else
    GOTO STEP 2
end if
STEP 2:
if C contiene un cero no cubierto **then**
    Encontrar un cero que no haya sido cubierto Z_0 y marcarlo con una tilde ($0^{'}$)
    if No hay ningún 0^* en la fila de Z_0 **then**
        GOTO STEP 3
    else
        Cubre esta fila, y descubre la columna conteniendo el 0^* **GOTO** STEP 2
    end if
else
    Guarda el elemento descubierto más pequeño e_{min}**. GOTO** STEP 4
end if
STEP 3: Construir la serie S alternado $0^{'}$y 0^* como sigue
Inserta Z_0 en S
while En la columna de Z_0 existe un $0^*(Z_1)$ **do**
    Inserta Z_1 en S
    Remplaza Z_0 con el $0^{'}$en la fila de Z_1. Inserta Z_0 en S
end while
Quita el asterísco de cada 0^* en S y remplaza todos los ceros con tilde por asteriscos. Borra todos los otros $0^{'}$ y descubre cada linea de C. **GOTO** STEP 1
STEP 4: Suma e_{min} a cada elemento de las filas cubiertas y restalo de cada elemento de las columnas descubiertas. **GOTO** STEP 2
DONE: Los pares de asignación se indican mediante las posiciones de 0^* en la matriz de coste

Para poder utilizar el algoritmo de Munkres en GED hay que tener en cuenta que la

matriz de coste C tiene que ser cuadrada. Por lo que hay que adaptar la matriz de coste definida en el problema de asignación. Esto es porque en el problema de asignación $\mathbf{C}_{ij}$ se correspondería con el coste de asignar el elemento i de g_1 con el elemento j de g_2 ,teniendo que ambos grafos tienen distinto número de enlaces en la mayoría de los casos. Para resolver esto, en [37] definen una nueva matriz de coste tal y como describo a continuación.

$$\mathbf{C} = \left[\begin{array}{cccc|cccc} c_{11} & c_{12} & \dots & c_{1\,m} & c_{1\,\epsilon} & \infty & \dots & \infty \\ c_{21} & c_{22} & \dots & c_{2\,m} & \infty & c_{2\,\epsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n\,1} & c_{n\,2} & \dots & c_{n\,m} & \infty & \dots & \infty & c_{n\,\epsilon} \\ - & - & - & - & - & - & - & - \\ c_{\epsilon\,1} & \infty & \dots & \infty & 0 & 0 & \dots & 0 \\ \infty & c_{\epsilon\,1} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \dots & \infty & c_{\epsilon\,m} & 0 & \dots & 0 & 0 \end{array}\right]$$

Definición. Matriz de coste para GED. $\mathbf{C}$ es la matriz de coste que define el problema de emparejamiento para dos grafos g1 y g2 con nodos $V_1 = \{u_1, \dots, u_n\}$y $V_2 = \{v_1, \dots, v_m\}$. En donde c_{ij}es el coste de sustitución $C(u_i \to v_j)$, $c_{i\epsilon}$es el coste de eliminación $C(u_i \to \epsilon)$ y $C_{\epsilon,j}$ el coste de inserción $C(\epsilon \to v_j)$. La matriz de coste está formada por cuatro cuadrantes. El de arriba a la izquierda representas los costes de sustitución de nodos de g1 por nodos de g2. El de arriba a la derecha los costes de eliminación de nodos en g1 y el de abajo a la izquierda los de inserción de nodos en g2. El cuadrante de abajo a la derecha es irrelevante porque no representa ninguna operación).

Con la matriz de coste $\mathbf{C}$ ya queda definido el problema de asignación por lo que se puede aplicar el algoritmo de Munkres 3. De esta manera el algoritmo encuentra el mínimo coste de asignación de los nodos de g1, representados por las filas, a los nodos de g2, columnas de la matriz $\mathbf{C}$. Cada asignación entre nodos es unívoca, es decir, un nodo de g1 solo podrá ser asignado a un nodo de g2 (cuadrante de arriba a la izquierda) o al nodo de eliminación ϵ (cuadrante de arriba a la izquierda). Del mismo modo cada nodo de g2 podrá ser asignado por un nodo de g1 en el cuadrante de arriba a la izquierda, o asignado al nodo de inserción ϵ (cuadrante de abajo a la izquierda).

En la definición de matriz de coste solo se tiene en cuenta el coste de los nodos y no se tiene en cuenta la información de los enlaces. Para lograr una mejor aproximación de la distancia verdadera de edición de grafos, es deseable implicar los costes de los enlaces en la asignación de los nodos. Para lograr este objetivo, se necesita una ampliación de la matriz de costes $\mathbf{C}$. Por ejemplo para la sustitución de nodos hay que añadir la suma mínima de las operaciones de edición de enlaces que implica la sustitución del nodo $u_i \to v_j$. Matemáticamente esta operación se expresa del siguiente modo. Si llamamos $E_{u\,i}$ al conjunto de los enlaces adyacentes al nodo u_i de g1 y $E_{v\,j}$ como el de los adyacentes al nodo $v_j$de g2 ; la suma

mínima equivale a resolver el problema de asignación del mismo modo que con la matriz **C** pero esta vez teniendo en cuenta unicamente los conjuntos de enlaces $E_{u\,i}$ e $E_{v\,j}$. En otras palabras, se crea una **matriz de coste de enlaces en la sustitución de nodos** con los dos conjuntos y se aplica el algoritmo de Munkres. De esta manera se establece una asignación óptima entre los elementos de $E_{u\,i}$ y $E_{v\,j}$. Lo que deriva en una suma mínima de operaciones de edición implicadas en la sustitución del nodo $u_i \rightarrow v_j$. Finalmente esta suma se añade al coste de sustitución c_{ij} en **C**. Igualmente para el coste de eliminación de un nodo u_i de g1, $c_{i\epsilon}$, se añade el coste de la eliminación de todos los enlaces adyacentes a u_i ; y para el coste de la inserción de un nodo v_j en g2, $C_{\epsilon,j}$, se añade el coste de todas las inserciones de los enlaces adyacentes de v_j.

Para finalizar es importante mencionar que el algoritmo de Munkres ofrece una solución óptima al problema de la asignación, pero nos proporciona una solución subóptima en el cálculo de la distancia de edición de grafos. Es decir, la distancia de edición obtenida es igual o mayor que el valor exacto de la distancia. Esto es debido a que en cada operación de edición de nodos únicamente se considera la estructura local del nodo. Expresado de otra manera, primero se computan las ediciones de los nodos y sobre los costes de estas ediciones se añaden los costes de las operaciones de edición de los enlaces. Por lo que no se tiene en cuenta el coste de editar los nodos y enlaces a la vez. Por ejemplo no se computa a la vez la operación de eliminar un enlace y todos sus nodos. Sino que primero se calcula la eliminación de los nodos y luego la de los enlaces.

3.3.3. Implementación

Como se ha comentado en la teoría de esta sección, para resolver GED como un problema de asignación se aplica Munkres sobre la matriz de costes. Por lo tanto, la construcción de la matriz y por ende el valor de los costes es la parte fundamental para que GED proporcione un emparejamiento adecuado. Ya que el algoritmo de Munkres da preferencia a los emparejamientos con menor coste. De los tres tipos de costes nodos: sustitución, inserción y eliminación; establezco un coste fijo o constante para las las operación de inserción y eliminación de nodos y enlaces debido a que es complicado estimar su valor. Se hace más hincapié en el coste de sustitución.

Además y debido a que los grafos que se utilizan en este trabajo tienen varios atributos, se van a generar varias matrices de coste. Por lo que es necesario normalizarlas para poder aplicarlas con éxito. Para normalizarlas se establece que todos los costes estén comprendidos entre 0 y 1.

Mi implementación de GED toma como punto de partida la elaborada por Pau Riba y Anjan Dutta [45] que a su vez está basada en el artículo de Bunke [37]. Implementan GED teniendo solo en cuenta las coordenadas de cada nodo. Por lo que he modificado su implementación original para permitir el cálculo de más atributos.

Resumiendo, se van a calcular las siguientes matrices de costes:

- Matriz de coste de los nodos:
  - Cuadrante de sustitución (arriba a la izquierda)
    - Suma de las matrices de coste sustitución de un nodo de g1 por un nodo de g2 debido al atributo:
      - Matriz de coste de Coordenadas
      - Matriz de coste de Betweenness
      - Matriz de coste de Degree
      - Matriz de coste de Vector_CM
      - Matriz de coste de Vector_Suma
      - Matriz de coste de Ángulos_nodo
  - Cuadrante de inserción (abajo a la izquierda)
    - Engloba el coste de inserción de un nodo de g2 y sus enlaces adyacentes
  - Cuadrante de eliminación (arriba a la derecha)
    - Engloba el coste de eliminación de un nodo de g1 y sus enlaces adyacentes
- Matriz de coste de los enlaces en la sustitución de nodos:
  - Se va a considerar como si fuese otro atributo de los nodos. La matriz se suma al cuadrante de sustitución de la matriz del coste de nodos.

En definitiva vamos a tener la matriz de coste de los nodos sobre la que se va a computar GED. En dicha matriz los cuadrantes de eliminación e inserción son fijos y el de sustitución va a ser la suma arbitraria de siete matrices de sustitución (seis debido a los seis atributos de los nodos y una debido a la matriz de coste de los enlaces en la sustitución de nodos. Al poder sumar arbitrariamente las matrices de sustitución podemos ver probar varias configuraciones. Y así determinar que combinación produce mejores asignaciones en GED.

La implementación de GED se divide en las siguientes etapas:

1. Cómputo de la matriz de coste de los nodos
2. Cómputo de la matriz de los enlaces en la sustitución de nodos
3. Grado de similitud y asignación de la correspondencia entre los nodos de g1 y g2

3.3.3.1. Cómputo de la matriz de coste de los nodos En esta etapa se aborda el cálculo de la matriz de coste de inserción, eliminación y sustitución de cada nodo. Lo primero de todo es definir los costes fijos de inserción y eliminación de nodos y enlaces. El valor de los costes es definido por ensayo y error:

- **Coste de eliminación de un nodo.** $C(u \to \epsilon) = 0{,}75$
- **Coste de inserción de un nodo.** $C(\epsilon \to v) = 0{,}75$
- **Coste de eliminación de un enlace.** $C(p \to \epsilon) = 0{,}25$
- **Coste de inserción de un enlace.** $C(\epsilon \to q) = 0{,}25$

El proceso de cálculo de la matriz de coste se realiza para los tres cuadrantes de la matriz de coste indicados en la definición 3.3.2.

- **Cuadrante de inserción de nodos:** Se rellena la diagonal con el coste de eliminación de cada nodo. Equivale a multiplicar el coste fijo de eliminación de un nodo por el número de enlaces adyacentes al nodo (E_{vj}).
  - $Cn_{\epsilon,j} = C(\epsilon \to v_j)\,|E_{v\,j}|$
- **Cuadrante de eliminación de nodos:** Se rellena la diagonal con el coste de inserción de cada nodo. Equivale a multiplicar el coste fijo de inserción de un nodo por el número de enlaces adyacentes al nodo (E_{ui}).
  - $Cn_{i,\epsilon} = C(u_i \to \epsilon)\,|E_{u\,i}|$
- **Cuadrante de sustitución de nodos:** $Cn_{i,j} = C(u_i \to v_j)$

  Para calcular el coste global de sustitución de los nodos, empezamos calculando el coste de sustitución de cada atributo por separado. Por lo que se crea una matriz de coste por cada atributo. De este modo se consigue solventar los problemas con la implementación de Pau Riba y Anjan Dutta [45]. El problema radica en que no se pueden computar todos los atributos juntos debido a que no todos los tienen la misma dimensión. Por ejemplo el atributo vector suma es de dimensión 2 y el degree tiene una única dimensión.

  El cálculo de la matriz de coste de sustitución es idéntico para todos los atributos salvo para el atributo Ángulos_nodo. En el resto, el coste de sustitución de un nodo u_i de g1 por el nodo v_j de g2 se obtiene hallando la distancia euclidiana entre los valores de g1 y g2 para un atributo en concreto. En general la distancia euclidiana $d_{i,j}$ entre u_i y v_j para un atributo de dimensión n , $A = \{A_1, A_2, \dots, A_n\}$,se define como $d_{i,j} = \sqrt{\sum_{k=1}^{n} {}^2(A_{i,k} - A_{j,k}}$. Por ejemplo para dos nodos u_i y v_j con Vector_Suma $(-25, 58)$ y $(-68, 56)$ la distancia euclidiana viene dada por $d_{i,j} = \sqrt{({}^2 -25 - (-68)) + ({}^2 58 - 56)} = \sqrt{1849 + 4} = 43{,}05$. Como se puede observar del ejemplo anterior, las distancias $d_{i,j}$ pueden variar mucho independientemente del atributo. Por lo que una vez que se ha obtenido la matriz de coste de sustitución para un atributo. Esta se normaliza escalando todos sus valores a un rango comprendido entre 0 y 1.

El cálculo de la matriz de coste de sustitución de nodos para el atributo Ángulos_nodo se realiza mediante el algoritmo 4. La idea del algoritmos es asignar un coste 0 entre aquellos nodos de g1 y g2 con ángulos similares dentro del atributo Ángulos_nodo. Es decir, por cada ángulo de Ángulos_nodo (para un nodo fijo de g1) se buscan las diferencias de ángulos, para todos nodos de g2, que son pequeñas y están cerca de un umbral. De esta manera, los nodos similares serán aquellos con más diferencias válidas (el contaje de las diferencias se calcula para cada nodo de g1). Por ejemplo, en la siguiente imagen se realiza el cálculo del coste para el mismo nodo de dos cubos similares. Se asigna un coste 0 entre los dos nodos porque entre ambos nodos hay 3 diferencias válidas si suponemos un umbral de 20^{o}:

- 270 - 270 = 0
- 180 - 180 = 0
- 45 - 30 = 15

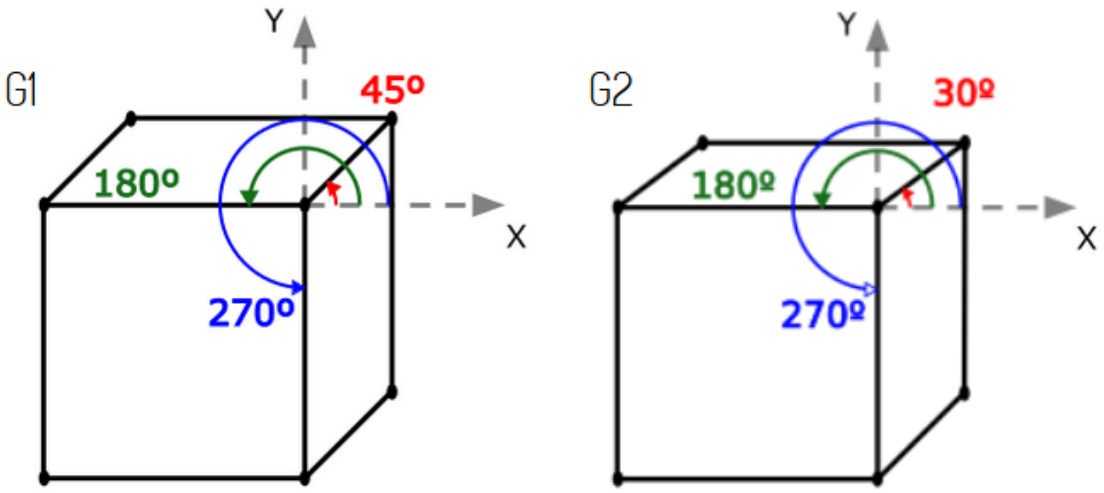


Fig. 33: Ejemplo del cálculo del coste del atributo Ángulos_nodo para el mismo nodo de dos cubos similares.

Algoritmo 4 Algoritmo de cálculo de la matriz de coste de sustitución de nodos para el atributo Ángulos_nodo.

1: **INPUTS:**
2: *G1_angs_nodos:* Lista con todos los atributos de *ángulos_nodo* de G1. Los nodos de la lista deben seguir el orden de los nodos en G1 (el mismo orden que para el resto de atributos de G1).
3: *G2_angs_nodos:* Lista con todos los atributos de *ángulos_nodo* de G2. Igualmente se tiene que mantener el orden de ocurrencia de los nodos en G2

umbral_ diff_ ang: Umbral para considerar válida la diferencia entre dos ángulos.
Coste_ OK: Coste de una asignación correcta. En este caso *Coste_ OK = 0.*
Coste_ NO: Coste de una asignación incorrecta. En este caso *Coste_ NO = 1.*
OUPUTS:
matriz_coste_ang_nodo: Matriz de coste de sustitución de un nodo u_i de G1 por un nodo v_j de G2. $c_{ij} = C(u_i \rightarrow v_j)$ para el atributo de *ángulos_ nodo.* Cada fila de *matriz_coste_ang_nodo* representa el coste del nodo u_i de G1 con cada nodo v_j de G2.

for each: *angs_ u_ i* de *G1_ angs_ nodos* **do**
    for each: *angs_ v_ j* de *G2_ angs_ nodos* **do**
        if *angs_ u_ i* no está vacía AND *angs_ v_ j* no está vacía **then**
            DIFF_ANGS_G1_G2(*angs_ u_ i, angs_ v_ j*)
            Recogemos en *lista_ diff_ angs_ Ui_ Vj* el retorno de **diff_angs_G1_G2** y lo añadimos a la lista *resultado_ diff_ angs_ Ui_ Vj*. Cada elemento de *lista_ diff_ angs_ Ui_ Vj* contiene las diferencias de todos los ángulos de *angs_ v_ j* con un ángulo de *angs_ u_ i.* Tendrá tantos elementos como ángulos haya en *angs_ u_ i*
        else
            Añadimos NO a la lista *resultado_ diff_ angs_ Ui_ Vj*
        end if
    end for
end for
for each: *lista_ diff_ angs_ Ui_ Vj* de *resultado_ diff_ angs_ Ui_ Vj* **do**
    Inicializamos *contador_ i_ j* a 0.
    if *lista_ diff_ angs_ Ui_ Vj* es distinta de NO **then**
        for *lista_ diff_ angs* de *lista_ diff_ angs_ Ui_ Vj* **do**
            Ponemos a False el *flag_ lista_ diff_ angs*
            for *diff_ ang* de *lista_ diff_ angs* **do**
                if valor_absoluto(*diff_ ang)* $\leq$ *umbral_ diff_ ang AND flag_ lista_ diff_ angs = False* **then**
                    Incrementamos en 1 *contador_ i_ j*
                    Ponemos a True el *flag_ lista_ diff_ angs*
                end if
            end for
        end for
    end if
    Añadimos *contador_ i_ j* a la lista *contadores_ angs_ validos_ i_ j*
end for
for *i* desde 0 hasta *número_ nodos_ G1 * número_ nodos_ G2* con paso *número_ nodos_ G2* **do**
    Recogemos en *contadores_ u_ i* todos los contadores de *contadores_ angs_ validos_ i_ j* para el *nodo_ i* . Es decir, los contadores desde el índice *i* hasta *i + número_ nodos_ G2.*
    Creamos *coste_ u_ i* poniendo el valor de *Coste_ OK* en aquellos índices de *contadores_ u_ i* donde el valor del contador sea máximo. Y ponemos el valor de *Coste_ NO*

en aquellos índices de *contadores_u_i* donde el valor del contador no sea máximo.
37: Añadimos *coste_u_i* a *matriz_coste_ang_nodo*
38: **end for**
39: **return** *matriz_coste_ang_nodo*

40: **function DIFF_ANGS_G1_G2**(*angs_u_i, angs_v_j*)
41: **for** each: *ang_u_i* de *angs_u_i* **do**
42: **for** each: *ang_v_j* de *angs_v_j* **do**
43: Calculamos *diff_ang = módulo(ang_u_i - ang_v_j + 180 + 360, 360) - 180.* Equivale a calcular la diferencia más pequeña entre dos ángulos. La valor de la diferencia está comprendido entre [-180º, 180º].
44: Añadimos a *lista_diff_angs* cada *diff_ang* calculada. La lista contiene las diferencias de cada ángulo de *angs_v_j* con *ang_u_i.*
45: **end for**
46: Añadimos a *lista_diff_angs_Ui_Vj* cada *lista_diff_angs* obtenida para cada *ang_u_i.*
47: **end for**
48: **return** *lista_diff_angs_Ui_Vj*
49: **end function**

3.3.3.2. Cómputo de la matriz de coste de los enlaces en la sustitución de nodos

A continuación se calcula la matriz de coste de los enlaces derivada de la sustitución de nodos. . En este caso se pretende obtener la matriz como si fuese un atributo más de los nodos. Es interesante recordar que los costes de los enlaces derivados de las operaciones de inserción y eliminación de nodos se han calculado en la sección anterior 3.3.3.1. Como se comentó en la definición de la matriz de coste de enlaces 3.3.2, los costes de los enlaces, se calculan en función de los enlaces adyacentes a un nodo i de g1 ($E_{u\,i}$) y de los enlaces adyacentes a un nodo j de g2 ($E_{v\,j}$). Es decir, se computa el problema de asignación entre los conjuntos de enlaces $E_{u\,i}$ y $E_{v\,j}$ pertenecientes al nodo i y j. Las etapas para realizar este cálculo se describen en los siguientes tres pasos:

1. **Cálculo de la matriz de coste de los enlaces derivados de la sustitución de un nodo i de g1 por un nodo j de g2:**
   a) **Coste de inserción de enlaces:** $Ce_{\epsilon,j} = C\left(\epsilon \rightarrow q_j\right)|E_{v\,j}|$**.** Se rellena la diagonal con el coste de eliminación de cada enlace. Equivale a multiplicar el coste fijo de eliminación de un enlace por el número de enlaces del conjunto E_{vj}.
   b) **Coste de eliminación de enlaces:** $Ce_{i,\epsilon} = C\left(p_i \rightarrow \epsilon\right)|E_{u\,i}|$**.** Se rellena la diagonal con el coste de inserción de cada enlace. Equivale a multiplicar el coste fijo de inserción de un enlace por el número de enlaces del conjunto E_{ui}.

   c) **Coste de sustitución de enlaces:** Calculo la distancia euclidiana de la longitud entre todos los enlaces de los conjuntos $E_{u\,i}$ y $E_{v\,j}$.

2. **Cálculo de la asignación:** Aplicación del algoritmo de Munkres

3. **GED de los enlaces, $d_{i,j}$:** Se obtiene sumando los valores de la matriz de coste de los enlaces para cada asignación del paso anterior.

Los tres pasos anteriores determinan la distancia $d_{i,j}$ que representa la distancia de edición derivada de la sustitución de un nodo i de g1 por un nodo j de g2. Si este mismo procedimiento se realiza sobre todos los nodos de g1 y g2 obtenemos la matriz de coste de los enlaces debido a la sustitución de todos los nodos de g1 y g2. La cuál ya puede ser tratada como si fuese la matriz de sustitución de nodos por un atributo como degree o coordenadas.

3.3.3.3. Grado de similitud y asignación de la correspondencia entre los nodos de g1 y g2

De la etapa anterior ya tenemos las matrices de costes necesarias para calcular el grado de similitud resolviendo el problema de asignación tal y como he planteado en 3.3.3.1. Pero, además, hay que realizar procesos de normalización previo a la aplicación de GED debido a que se van a sumar matrices cuyos valores son muy dispares entre sí. Divido este proceso en las siguientes etapas:

1. **Primera normalización de las matrices de costes.** En esta etapa se realiza la normalización de cada matriz de coste. Por lo que ahora las matrices de sustitución de cada atributo tienen sus valores comprendidos entre 0 y 1.

2. **Suma de matrices de costes .** Aquí es donde se crea la matriz de coste global de g1 y g2. Se suman la matriz con los costes de inserción y eliminación junto con las matrices de sustitución de los atributos que tengo en cuenta.

3. **Segunda normalización de las matrices de costes.** Es necesario normalizar la matriz de coste global ya que al sumar las matrices de los atributos, la matriz de coste global cambia y sus valores ya no están comprendidos entre 0 y 1.

4. **Grado de similitud y asignación de la correspondencia entre los nodos de g1 y g2.**

   a) **Cálculo de la asignación:** Aplicación del algoritmo de Munkres

   b) **GED entre g1 y g2:** La obtengo sumando los valores de la matriz de coste para cada asignación del paso anterior.

3.3.4. Implementación de la evaluación de las asignación de GED

En este apartado implemento la evaluación supervisada de las asignaciones entre nodos de g1 y g2 devueltas por GED. En concreto he etiquetado un número fijo de puntos que considero relevantes en cada tipo de dibujo. He ido asignando cada punto n del modelo con el punto n que creo que coincide con el dibujo del paciente. Es decir, por ejemplo si considero que el punto 1 del triangulo modelo es el vértice de abajo a la izquierda, asigno como punto 1 del triangulo del paciente el mismo vértice. Esto no es problema para los dibujos con poca distorsión pero a veces no he encontrado una correspondencia clara entre puntos. En estos casos, marco que la etiquetación del punto n no es válida y no la tengo en cuenta en el cálculo del ratio de correspondencia. En el caso de etiquetar una asignación correcta se marca cada punto con el botón izquierdo del ratón, y con el derecho cuando no hay una asignación clara. En el caso de que se etiqueta con el botón derecho, se cambian las coordenadas marcadas por otras que se encuentran fuera del dibujo por lo que cuando se calcula la correspondencia ese punto no se tiene en cuenta. Para que se puedan sacar conclusiones válidas he etiquetado los dibujos de tres pacientes por cada clase lo que se traduce en aproximadamente un 23 % de todos los pacientes. Para etiquetar los puntos he desarrollado una interfaz de usuario (GUI) con la aplicación GUIDE de Matlab.

A continuación voy a mostrar varias imágenes mostrando el proceso de etiquetación:

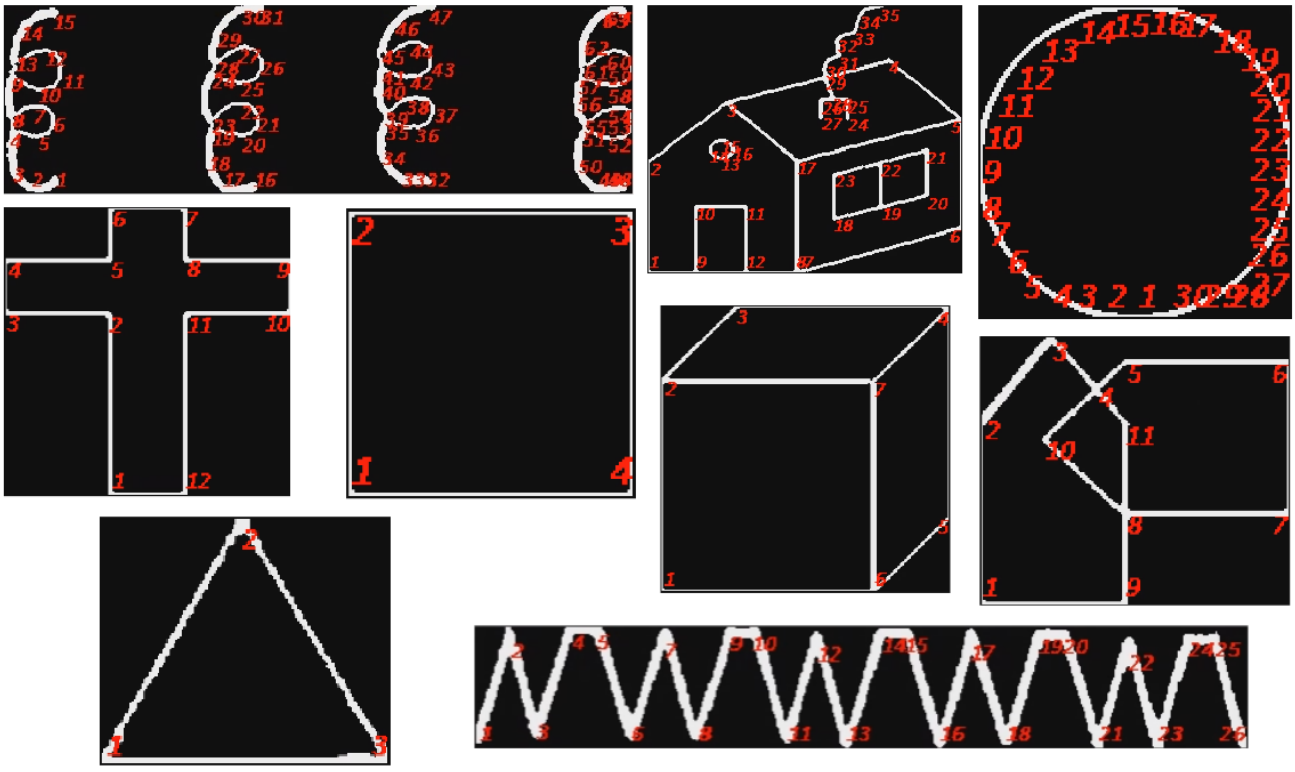


Fig. 34: Etiquetación de todos los modelos.
64 puntos etiquetados en los bucles, 35 en la casa, 30 en el círculo, 12 en la cruz, 4 en el cuadrado, 7 en el cubo, 11 en el minimental, 26 en los picos-mesetas y 3 en el triangulo.

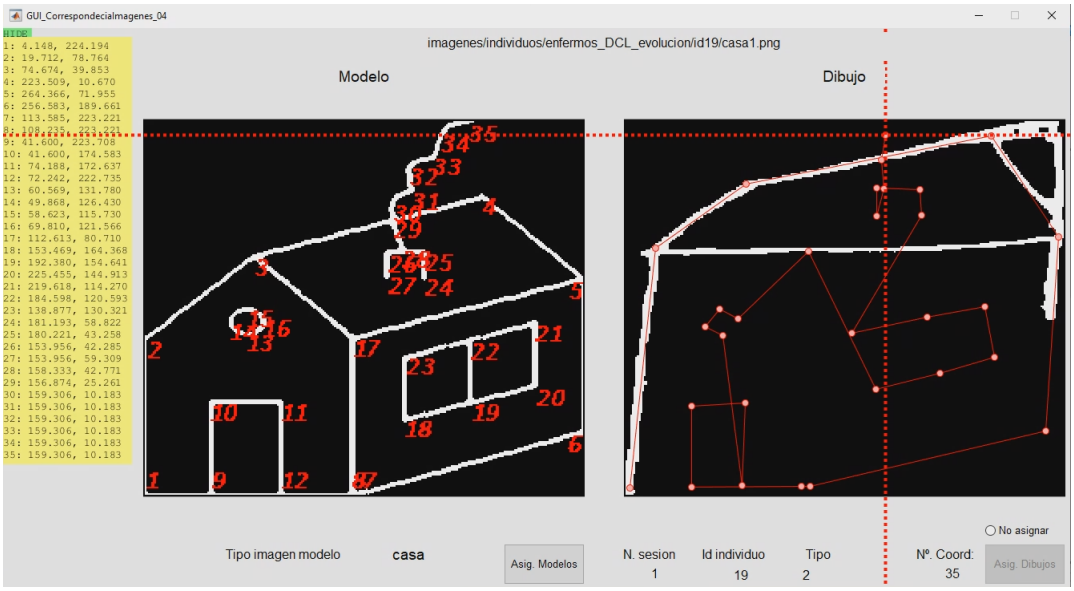


Fig. 35: Etiquetación de una casa en la que faltan bastantes regiones.

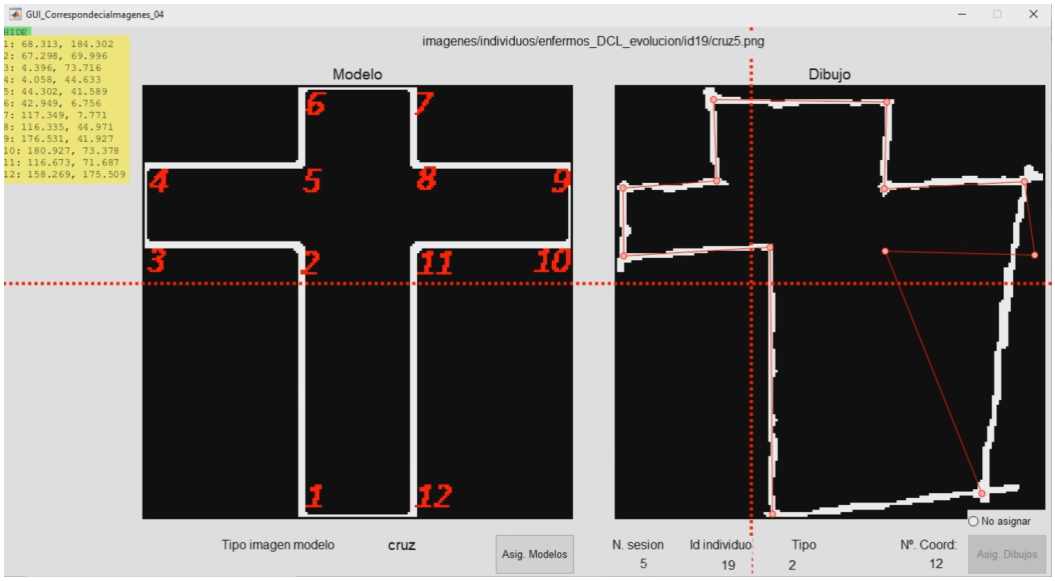


Fig. 36: Etiquetación de una cruz en la que faltan algún punto.

Para calcular la evaluación una vez que se tiene las asignación entre g1 y g2. Analizo iterativamente si se encuentra cada asignación etiquetada. Primero obtengo los indices de los nodos de g1 y g2 cercanos al punto de asignación etiquetado (tanto para el grafo modelo como para el grafo del paciente). Para evaluar la cercanía he establecido un umbral con un valor de 15 (obtenido por ensayo y error). Una vez que se que los nodos cercanos, compruebo si GED a devuelto una asignación entre cualquiera de los puntos cercanos de g1 y g2. De este modo ya tengo todas las asignaciones de GED que son válidas dado el umbral.

Es verdad, que para una asignación etiquetada cualquiera puede haber múltiples asignaciones de GED que cumplan la condición del umbral, e igualmente puede darse la circunstancia de que una misma asignación de GED cumpla las condiciones de varias asignaciones etiquetadas. Para ello, calculo las asignaciones únicas de GED seleccionando aquellas cuyas suma de distancias (distancia entre nodo de g1 y el nodo etiquetado e igualmente para g2) sea mínima.

3.4. Métricas para el diagnóstico del DCL

Finalmente ya hemos terminado de calcular la distancia GED que transforma g1 en g2. A su vez he calculado dos métricas proporcionadas por la asignación. Paso a enumerar las métricas obtenidas en este módulo:

- **Grado de similitud.** da una medida la distancia GED que supone transformar el grafo modelo en el grafo del paciente. Si la vale cero significará que ambos grafos son

completamente idénticos y cuanto mayor sea la distancia , mayor será la diferencia entre los grafos.

- **Número de nodos eliminados de g1.** Esta variable contabiliza el número de nodos de g1 que han sido eliminados al no encontrarse un nodo equivalente en g2. Supongo que los dibujos de pacientes en los que faltan bastantes regiones a simple vista deberían mostrar un valor alto en comparación con los dibujo que contienen la mayor parte de las regiones de la imagen.
- **Número de nodos insertados en g2.** Contabiliza el número de nodos de g2 que han sido insertados en g1 por no encontrarse una correspondencia en g1 de ese nodo de g2. Creo que este valor será alto en los dibujos de los pacientes que tienen muchas distorsiones como trazos quebrados o borrones, etc.

4. Resultados

4.1. Resultados del preprocesamiento de las imágenes

A continuación muestro algunos ejemplos del preprocesamiento de las imágenes. En general se ha conseguido atenuar bastante el ruido, ya sea el ruido de la digitalización o el gaussiano, aunque en casos en el que la presencia de ruido es importante, la atenuación no ha sido la misma. Por ejemplo, en la siguiente figura 37 se muestran cuatro ejemplos de imágenes con ruido de digitalización (imagen original a la izquierda y preprocesada a la derecha. En **A** y **B** el ruido de digitalización es importante (se puede observar como aparecen los agujeros del papel) y aunque el ruido se ha reducido, este sigue apareciendo en la imagen preprocesada por lo que el proceso de Graph Matching tiene que ser robusto a este ruido. Otro ejemplo con ruido de digitalización es **D**, en este caso hay una línea horizontal en la parte inferior que al tener un tamaño apreciable (contiene píxeles que el umbral de filtrado) no ha podido ser eliminado. Por último en **C**, el fondo de la imagen original no es completamente blanco por lo que esto ha afectado a la umbralización provocando que el ruido gaussiano no haya sido eliminado de las misma manera que si ha lo ha sido en imágenes originales con nivel similar de ruido gaussiano. Pero como indico al inicio del párrafo, la mayor parte de las imágenes si han sido bien preprocesadas y se ha conseguido reducir en una gran medida el ruido presente.

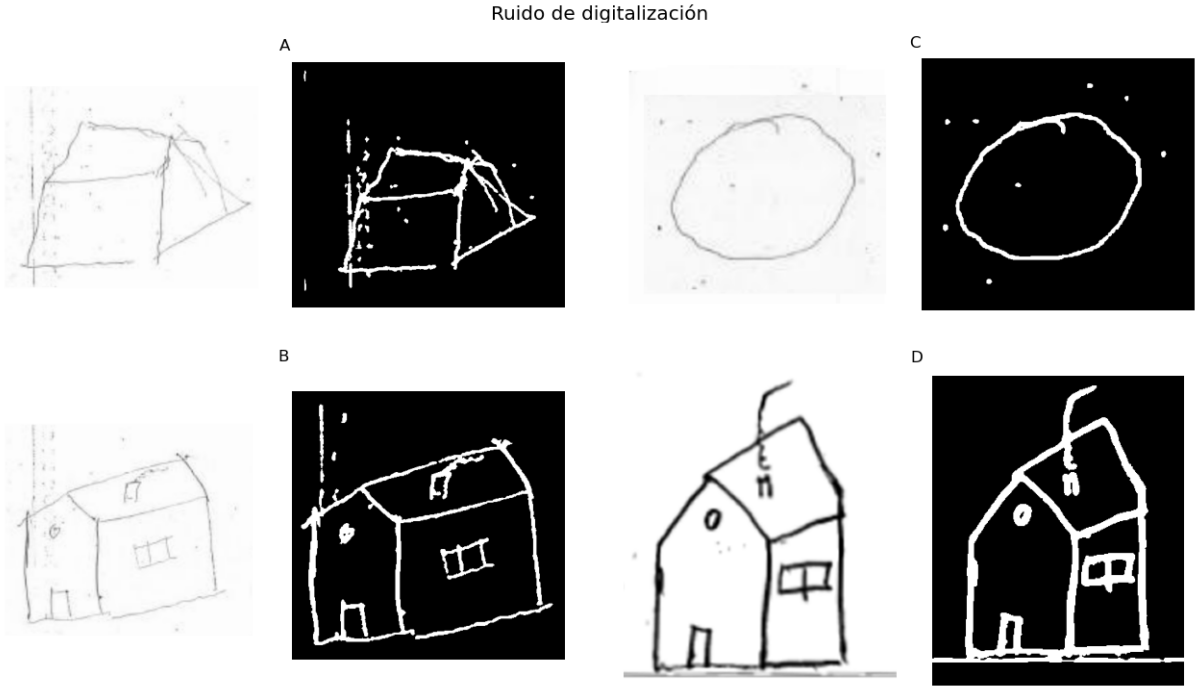


Fig. 37: Ejemplos de imágenes preprocesadas (original izquierda y preprocesada a la derecha) con ruido de digitalización.

En cuánto a los trazos débiles, la umbralización de Otsu en niveles dos o tres permite procesar los trazos. Aunque algunas veces aparecen bastante fragmentados por lo que algún fragmento ha sido eliminado en el filtrado por áreas. La umbralización en tres niveles consigue digitalizar mejor los trazos débiles que la de dos niveles tal y como se aprecia la figura 38 (círculos rojos). Esto propicia que mejore la continuidad de los trazados facilitando que mejore su segmentación.

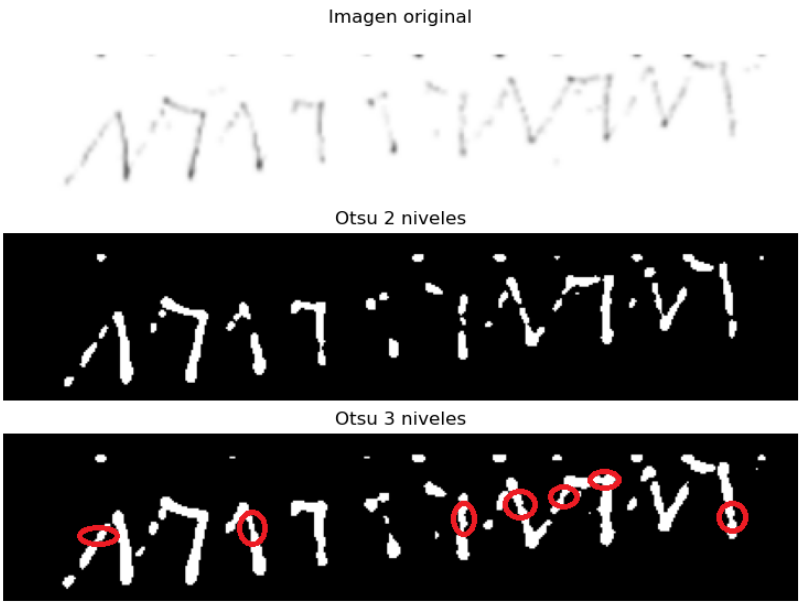


Fig. 38: Umbralización de Otsu con dos y tres niveles para un dibujo con trazos débiles. Se ha indicado con círculos rojos las zonas con trazos débiles que mejor han sido digitalizadas por la de tres niveles en comparación con la de dos.

En la figura 39 se observa el preprocesamiento de cuatro imágenes con trazos débiles. Las imágenes **A**, **B** y **C** han sido procesadas adecuadamente y se puede apreciar que aparecen la mayoría de trazos débiles. Aún así hay algunos trazos que se recogen. En la imagen **D** vemos un ejemplo más claro de esto último, la líneas interiores del cubo no están bien recogidas al ser algo difusas y al estar el trazo punteado. Es un caso similar al de la imagen de los picos-mesetas de la figura 38 porque si la umbralización de los trazos difusos origina segmentos rotos, es fácil que se eliminen si su área es inferior al umbral.

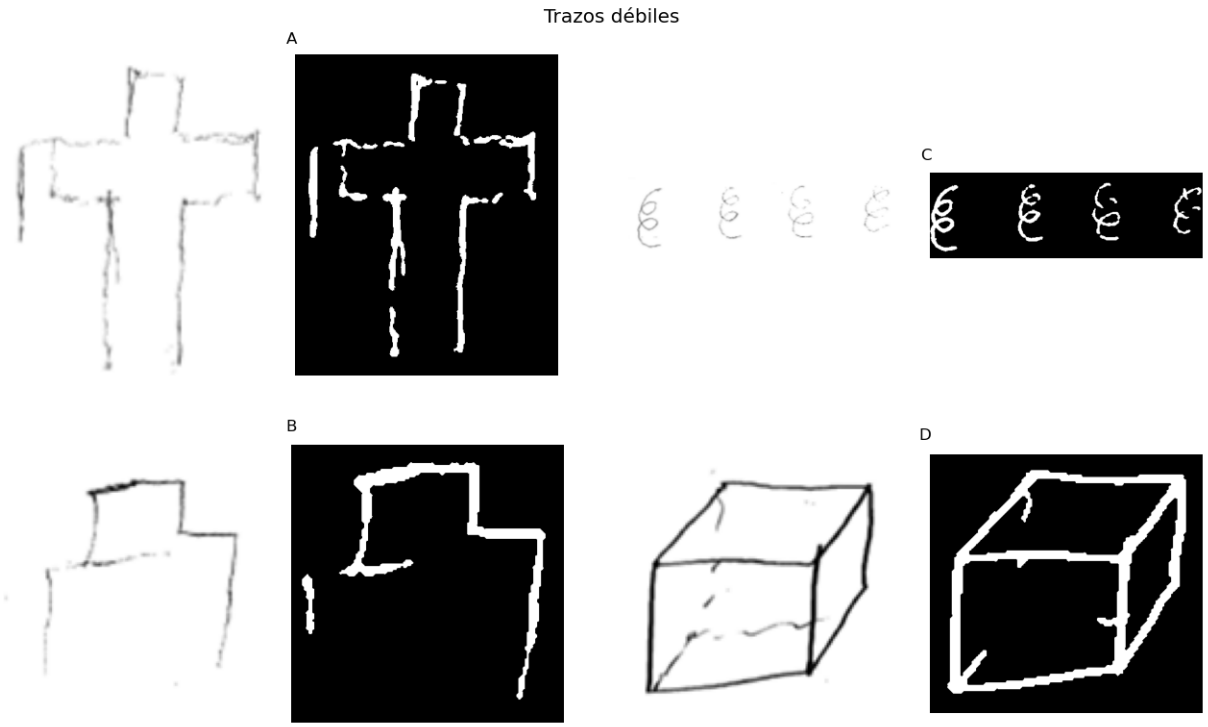


Fig. 39: Ejemplos de imágenes preprocesadas (original izquierda y preprocesada a la derecha) con trazos débiles.

Seguidamente muestro otras cuatro imágenes (figura 40) que resumen como ha sido el preprocesamiento implementado en imágenes con poca cantidad de ruido o sin trazos débiles. Como comento al inicio de este apartado, el preprocesamiento ha sido bueno en la mayoría de las imágenes, aunque ha habido ocasiones en dónde se han distorsionado algo los trazos originales. Por ejemplo, la imagen original del cubo **B** tiene muchos trazos pequeños y muy juntos, los cuáles se han perdido en la imagen preprocesada ya que todos se unen en una línea más gruesa. Algo similar sucede con **C** (círculo rojo), ya que varios picos y mesetas aparecen muy juntos a pesar de que en la original si hay separación. Este comportamiento no se aprecia tanto en **A** y **D**. Es verdad que hay trazos juntos en algunos rizos de los bucles, pero esto mismo no sucede en muchas otras líneas que están separadas por una distancia similar a la de los picos y mesetas del círculo rojo de **C**. Esto sucede durante el proceso de escalado debido a la diferencia de tamaños entre la imagen original de **C** y la imagen modelo de los picos-mesetas (185 x 57 frente a 536 x 100). Por tanto, el algoritmo de escalado ha interpolado bastante por la falta de información.

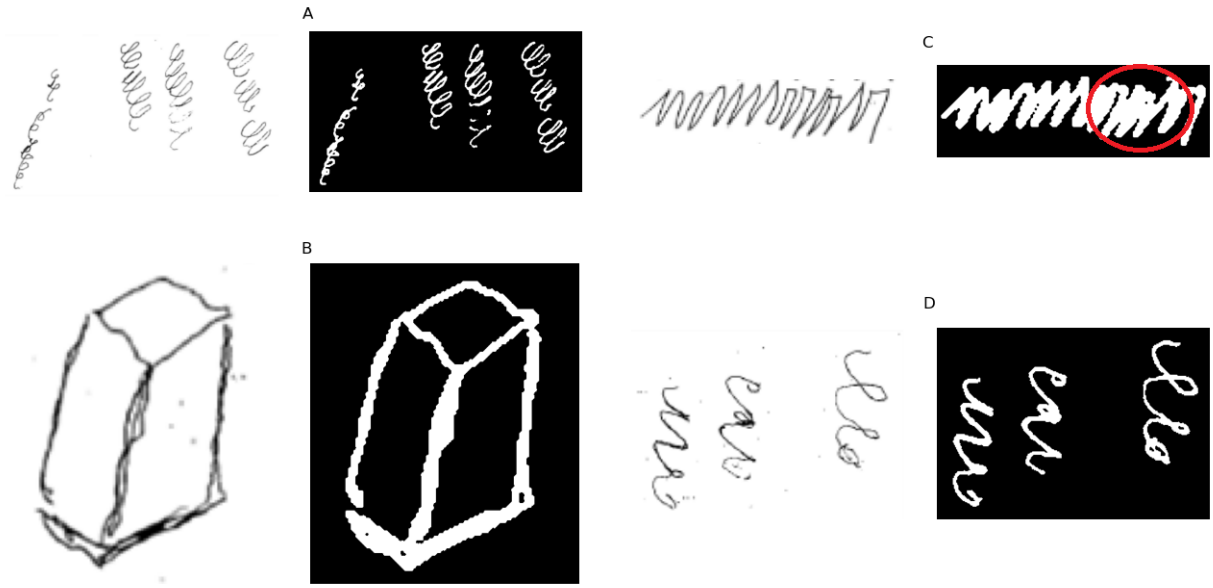


Fig. 40: Ejemplo de transformaciones realizadas en el preprocesamiento de las imágenes con poca cantidad de ruido o sin trazos débiles.

4.1.1. Conclusiones

En general se han conseguido buenos resultados en el preprocesamiento de las imágenes originales. Aproximadamente el 90 % han sido preprocesadas adecuadamente. Y el resto tienen distorsiones como las mostradas anteriormente que no deberían repercutir en exceso en las siguientes etapas. Los errores con los trazos débiles pueden que sean las distorsiones que originen más problemas en el Graph Matching; ya que al haber más contornos desconectados se pueden crear más nodos y enlaces con poco valor informativo. De todas maneras el proceso de Graph Matching debe de ser lo suficientemente potente para tratarlas adecuadamente. Además la umbralización en tres niveles es adecuada para digitalizar los trazos débiles sin incurrir en añadir un ruido excesivo.

4.2. Resultados de las aproximaciones de contornos

En esta parte voy a exponer los resultados de las aproximaciones de contornos tanto para la aproximación con esquinas Harris como de las aproximaciones con RDP. He aplicado las dos aproximaciones a todas las imágenes etiquetadas en la evaluación de las asignaciones. Para medir la exactitud de las aproximaciones utilizo el coeficiente de Dice entre las imágenes preprocesadas y las aproximadas. Este coeficiente proporciona un valor entre 0 y 1 indicando con 1 si las dos imágenes están completamente superpuestas y 0 si no lo están en absoluto. La formula del coeficiente de Dice utilizado es la siguiente (únicamente se tienen en cuenta los píxeles del blancos, es decir, los del frente):

$$Dice = \frac{2 * Número\ píxels\ imagen\ aproximada}{Número\ píxels\ imagen\ aproximada \quad + \quad Número\ píxels\ imagen\ preprocesada}$$

En la siguiente tabla se muestra el promedio de los coeficientes de Dice según el tipo de dibujo y de paciente:

Dibujo	**Aprox. con esquinas harris**	**Aprox. con RDP**
Bucles sanos	0.75	0.75
Bucles DCL Estable	0.81	0.81
Bucles DCL Evolución	0.73	0.73
Casas sanos	0.68	0.75
Casas DCL Estable	0.67	0.74
Casas DCL Evolución	0.67	0.7
Círculos Sanos	0.88	0.88
Círculos DCL Estable	0.81	0.81
Círculos DCL Evolución	0.7	0.7
Cruz Sanos	0.66	0.72
Cruz DCL Estable	0.71	0.77
Cruz DCL Evolución	0.66	0.71
Cuadrado Sanos	0.64	0.79
Cuadrado DCL Estable	0.63	0.82
Cuadrado DCL Evolución	0.58	0.75
Cubo sanos	0.66	0.68
Cubo DCL Estable	0.66	0.68
Cubo DCL Evolución	0.64	0.67
Minimental Sanos	0.68	0.77
Minimental DCL Estable	0.68	0.76
Minimental DCL Evolución	0.71	0.8
Picos-mesetas Sanos	0.63	0.65
Picos-mesetas DCL Estable	0.64	0.64
Picos-mesetas DCL Evolución	0.65	0.62
Triángulos Sanos	0.69	0.72
Triángulos DCL Estable	0.66	0.8
Triángulos DCL Evolución	0.71	0.76
Promedio	**0.69**	**0.74**

Tab. 2: Resultados de la evaluación de las aproximaciones con esquinas Harris y RDP según el tipo de dibujo y de paciente.

Y la misma tabla de manera resumida según el tipo de dibujo:

Dibujo	Aprox. con esquinas harris	Aprox. con RDP
Bucles	0.76	0.76
Casas	0.67	0.73
Círculos	0.8	0.8
Cruz	0.68	0.73
Cuadrado	0.62	0.79
Cubo	0.65	0.68
Minimental	0.69	0.78
Picos-mesetas	0.64	0.64
Triángulos	0.69	0.76
Promedio	**0.69**	**0.74**

Tab. 3: Resultados de la evaluación de las aproximaciones con esquinas Harris y RDP según el tipo de dibujo.

4.2.1. Conclusiones

En general tanto utilizando las esquinas Harris como RDP se consiguen buenas aproximaciones (0.69 frente a 0.74 en el promedio del coeficiente de Dice). Aunque en es verdad que hay una ligera diferencia en favor de la RDP en la mayoría de las imágenes (los bucles y círculos obtienen la misma puntuación porque ambos se aproximan con RDP tal y como se indicó en 3.2.3.1) Esta diferencia es más notable en los cuadrados y minimentals. Una posible explicación es que la aproximación con esquinas Harris ha simplificado más aquellos intervalos del contorno donde las esquinas Harris están más distanciadas y por tanto más pixeles en la de Harris que quedan fuera del contorno original. Por ejemplo esto lo podemos apreciar en la siguiente imagen (A es la imagen preprocesada con las esquinas Harris en rojo, B la aproximación de Harris y C la de RDP). Vemos como el contorno dentro de la elipse azul es aproximada por un segmento en la aproximación de Harris y por dos en la de RDP.

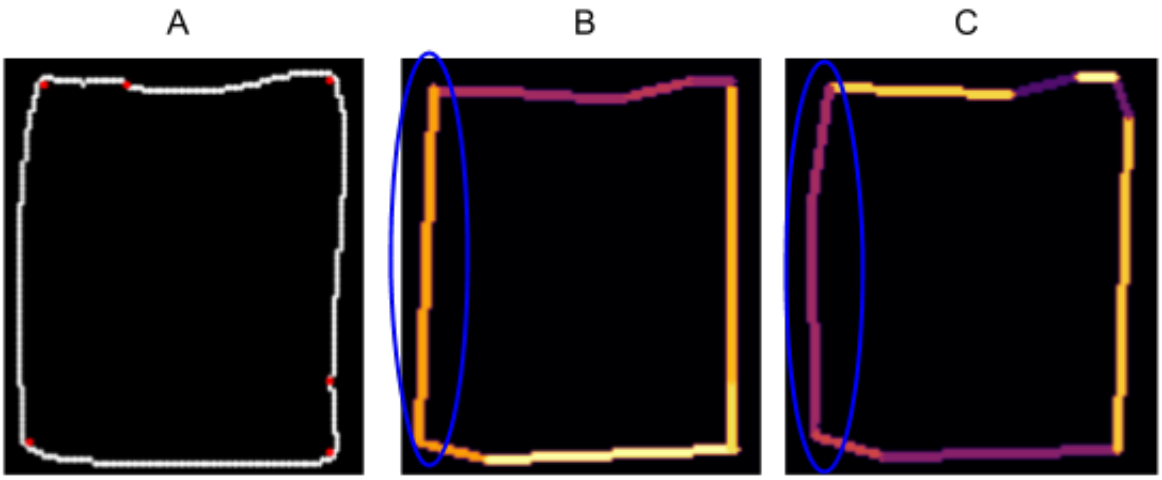


Fig. 41: Ejemplos de la simplificación de los contornos de los círculos.

4.3. Resultados de la evaluación de las asignaciones

Como he comentado en la sección 3.3.4 se han etiquetado un 23 % de las imágenes. De esta manera vamos a disponer de una base suficiente para analizar varias configuraciones de la matriz de coste del algoritmo GED y así ver cual es la que ha emparejado mejor los nodos. Una vez hallada la mejor configuración, se utiliza esta para el diagnóstico de DCL. A su vez se van a comparar las dos aproximaciones de contornos para evaluar cuál consigue mejores asignaciones..

Se va a probar el comportamiento de las matrices de costes de los atributos. Ya sea por separado o combinados. En todas las configuraciones se utilizan las mismas matrices de inserción y eliminación. Salvo en la configuración H que únicamente se utilizan las matrices de coste de inserción y eliminación para ver su influencia en el emparejamiento. Las configuraciones de atributos probadas son:

- A: Matriz de coste de sustitución - atributo de Coordenadas.
- B: Matriz de coste de sustitución - atributo de Betweennes.
- C: Matriz de coste de sustitución - atributo de Degree.
- D: Matriz de coste de sustitución - atributo de Vector_CM.
- E: Matriz de coste de sustitución - atributo de Vector_Suma.
- F: Matriz de coste de sustitución - atributo de Ángulos_Nodo.
- G: Matriz de coste de sustitución - atributo de Enlaces.

- H: Únicamente matrices de inserción y eliminación.
- I: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM y Ángulos_Nodo.
- J: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM, Vector_Suma y Ángulos_Nodo.
- K: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM y Vector_Suma.
- L: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM, Vector_Suma, Ángulos_Nodo y Degree.
- M: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM, Vector_Suma, Ángulos_Nodo y Enlaces.
- N: Todas las matrices de coste

4.3.1. Resultados de las evaluaciones para la aproximación con esquinas Harris

En la siguiente tabla se muestra el promedio de las asignaciones para la aproximación con esquinas Harris según el tipo de dibujo y de paciente para todas las configuraciones únicas:

Dibujo	A	B	C	D	E	F	G	H
Bucles sanos	77.46	8.66	13.18	78.08	8.35	17.36	4.78	10.76
Bucles DCL Estable	55.36	9.82	7.44	48.51	7.74	7.44	5.06	8.63
Bucles DCL Evolución	61.48	9.56	8.36	55.55	6.56	10.75	4.17	12.53
Casas sanos	62.9	14.77	16.8	57.49	28.7	41.4	9.91	7.44
Casas DCL Estable	51.71	15.9	17.28	52.5	18.97	39.9	6.48	5.88
Casas DCL Evolución	35.93	5.93	17.46	34.33	12.77	23.63	5.32	9.69
Círculos Sanos	87.5	59.38	59.38	87.5	38.54	85.42	26.04	59.38
Círculos DCL Estable	81.25	43.75	43.75	81.25	37.5	67.71	20.83	43.75
Círculos DCL Evolución	83.33	47.92	47.92	83.33	33.33	66.67	19.79	41.67
Cruz Sanos	100	22.22	29.17	97.22	69.44	95.83	15.28	29.17
Cruz DCL Estable	90.28	12.5	12.5	90.28	81.94	100	6.95	12.5
Cruz DCL Evolución	90.28	8.33	8.33	69.44	65.28	93.06	11.11	9.72
Cuadrado Sanos	100	8.33	8.33	100	95.83	95.83	16.67	12.5
Cuadrado DCL Estable	100	25	25	95.83	87.5	100	25	25
Cuadrado DCL Evolución	95.83	8.33	8.33	95.83	95.83	95.83	12.5	8.33
Cubo sanos	80.95	9.53	28.57	73.81	42.86	83.33	14.29	7.15
Cubo DCL Estable	78.57	7.15	28.58	83.33	35.72	54.76	26.19	11.91
Cubo DCL Evolución	82.14	12.3	4.76	71.03	40.48	69.05	9.92	12.3
Minimental Sanos	80	28.79	40	76.97	56.97	84.7	13.94	25.91
Minimental DCL Estable	88.55	16.91	29.82	90.55	50.54	88.55	15.09	14.91
Minimental DCL Evolución	90.91	18.18	36.36	84.85	30.3	78.79	6.06	18.18
Picos-mesetas Sanos	85.9	12.82	4.49	65.39	3.21	21.79	4.49	2.56
Picos-mesetas DCL Estable	70.31	12.9	3.9	55.13	5.82	19.33	1.95	4.57
Picos-mesetas DCL Evolución	45.95	10.31	3.87	34.72	4.04	11.12	3.42	1.28
Triángulos Sanos	100	38.89	27.78	94.45	94.45	94.45	33.33	27.78
Triángulos DCL Estable	100	33.33	38.89	94.45	88.89	94.45	44.44	44.44
Triángulos DCL Evolución	94.45	11.11	16.67	88.89	88.89	94.45	22.22	44.44
Promedio (%)	**80.41**	**18.99**	**21.74**	**75.58**	**45.57**	**64.28**	**14.27**	**18.98**

Tab. 4: Resultados de la evaluación de las asignaciones de GED para la aproximación con esquinas Harris utilizando únicamente una matriz de coste

En la siguiente tabla se muestra el promedio de las asignaciones según el tipo de dibujo y de paciente para todas las configuraciones con varias matrices de coste:

Dibujo	I	J	K	L	M	N
Bucles sanos	69.44	67.03	74.21	60.72	63.12	59.85
Bucles DCL Estable	42.26	39.58	54.76	36.91	40.18	37.2
Bucles DCL Evolución	52.53	48.98	52.82	46.59	47.19	43.9
Casas sanos	70.62	74.65	67.25	70.68	75.64	72.7
Casas DCL Estable	73.47	74.97	66.91	70.96	75.46	74.41
Casas DCL Evolución	40.31	43.4	43.84	38.09	44.35	38.22
Círculos Sanos	87.5	86.46	84.38	86.46	86.46	86.46
Círculos DCL Estable	80.21	79.17	78.13	79.17	79.17	79.17
Círculos DCL Evolución	86.46	78.13	73.96	78.13	81.25	82.29
Cruz Sanos	100	100	98.61	98.61	100	98.61
Cruz DCL Estable	100	100	98.61	100	100	100
Cruz DCL Evolución	98.61	97.22	90.28	100	100	100
Cuadrado Sanos	100	100	100	95.83	100	95.83
Cuadrado DCL Estable	100	100	100	100	100	100
Cuadrado DCL Evolución	95.83	100	100	95.83	100	95.83
Cubo sanos	85.71	88.09	85.71	88.09	90.47	83.33
Cubo DCL Estable	80.95	78.57	88.09	90.47	73.81	80.95
Cubo DCL Evolución	83.73	89.29	84.92	84.13	86.51	73.81
Minimental Sanos	90.91	89.24	89.24	89.24	89.24	90.91
Minimental DCL Estable	98	100	92.36	100	98.18	100
Minimental DCL Evolución	93.94	96.97	93.94	93.94	96.97	87.88
Picos-mesetas Sanos	96.15	96.15	73.08	94.23	96.15	92.31
Picos-mesetas DCL Estable	83.13	84.49	57.85	78.64	85.13	76.03
Picos-mesetas DCL Evolución	56.4	57.68	46.45	57.04	57.68	54.5
Triángulos Sanos	100	100	94.45	100	100	100
Triángulos DCL Estable	100	100	100	100	100	94.45
Triángulos DCL Evolución	100	100	100	100	100	94.45
Promedio (%)	**83.93**	**84.08**	**81.11**	**82.73**	**83.96**	**81.23**

Tab. 5: Resultados de la evaluación de las asignaciones de GED para la aproximación con esquinas Harris combinando varias matrices de matriz de coste

4.3.1.1. Mejores configuraciones

- J: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM, Vector_Suma y Ángulos_Nodo.

- M: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM, Vector_Suma, Ángulos_Nodo y Enlaces.
- I: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM y Ángulos_Nodo.

4.3.2. Resultados de las evaluaciones para la aproximación con RDP

En la siguiente tabla se muestra el promedio de las asignaciones para la aproximación con RDP según el tipo de dibujo y de paciente para todas las configuraciones únicas:

Dibujo	A	B	C	D	E	F	G	H
Bucles sanos	77.46	8.66	13.18	48.51	8.35	13.16	4.78	8.63
Bucles DCL Estable	55.36	9.82	7.44	55.55	7.74	6.25	5.06	12.53
Bucles DCL Evolución	61.48	9.56	8.36	55.39	6.56	5.09	4.17	5.91
Casas sanos	60.36	9.85	6.89	49.51	22.24	18.33	9.86	8.03
Casas DCL Estable	53.52	8.95	6.57	35.51	12.07	20.69	6.94	9.09
Casas DCL Evolución	37.84	4.83	11.08	87.5	17.17	10.6	6.33	59.38
Círculos Sanos	87.5	59.38	59.38	81.25	38.54	87.5	26.04	43.75
Círculos DCL Estable	81.25	43.75	43.75	83.33	37.5	77.08	20.83	41.67
Círculos DCL Evolución	83.33	47.92	47.92	98.61	33.33	73.96	19.79	43.06
Cruz Sanos	100	43.06	45.83	98.61	90.28	72.22	15.28	16.67
Cruz DCL Estable	94.45	18.06	18.06	83.33	77.78	88.89	8.33	23.61
Cruz DCL Evolución	95.84	20.84	26.39	100	73.61	90.28	18.06	25
Cuadrado Sanos	100	16.67	16.67	95.83	100	91.67	25	20.83
Cuadrado DCL Estable	100	20.83	25	95.83	100	87.5	20.83	33.33
Cuadrado DCL Evolución	95.83	33.33	33.33	97.62	95.83	87.5	29.17	14.29
Cubo sanos	97.62	19.05	9.53	80.95	57.14	45.24	9.53	23.81
Cubo DCL Estable	78.57	11.91	4.76	75	47.62	35.72	11.91	25.79
Cubo DCL Evolución	82.94	28.57	18.65	92.27	52.38	29.37	10.72	30.6
Minimental Sanos	89.24	15.45	29.39	92.36	70.61	70.91	33.64	26.18
Minimental DCL Estable	92.36	42.73	41.09	87.88	59.28	72.18	7.45	24.24
Minimental DCL Evolución	90.91	27.27	30.3	67.31	60.61	60.61	12.12	3.21
Picos-mesetas Sanos	76.92	12.18	7.69	36.05	12.82	10.9	2.57	5.23
Picos-mesetas DCL Estable	59.95	6.44	7.79	38	7.16	11.75	3.85	4.49
Picos-mesetas DCL Evolución	44.22	9.46	8.77	100	13.17	7.46	3.4	38.89
Triángulos Sanos	94.45	33.33	33.33	94.45	83.34	66.67	38.89	44.44
Triángulos DCL Estable	100	44.44	38.89	88.89	94.45	66.67	50	61.11
Triángulos DCL Evolución	94.45	33.33	38.89	88.89	83.33	61.11	27.78	44.44
Promedio (%)	**80.96**	**23.69**	**23.66**	**78.09**	**50.48**	**50.72**	**16.01**	**25.86**

Tab. 6: Resultados de la evaluación de las asignaciones de GED para la aproximación RDP utilizando únicamente una matriz de coste

En la siguiente tabla se muestra el promedio de las asignaciones según el tipo de dibujo y de paciente para todas las configuraciones con varias matrices de coste:

Dibujo	**I**	**J**	**K**	**L**	**M**	**N**
Bucles sanos	69.44	67.03	74.21	60.72	63.12	59.85
Bucles DCL Estable	42.26	39.58	54.76	36.91	40.18	37.2
Bucles DCL Evolución	52.53	48.98	52.82	46.59	47.19	43.9
Casas sanos	58.4	59.97	65.33	56.44	60.94	60.87
Casas DCL Estable	59.99	62.65	64.54	58.17	64.12	58.19
Casas DCL Evolución	33.77	35.39	46.6	31.69	37.61	33.37
Círculos Sanos	87.5	86.46	84.38	86.46	86.46	86.46
Círculos DCL Estable	80.21	79.17	78.13	79.17	79.17	79.17
Círculos DCL Evolución	86.46	78.13	73.96	78.13	81.25	82.29
Cruz Sanos	100	100	100	95.83	100	90.28
Cruz DCL Estable	100	100	100	100	100	100
Cruz DCL Evolución	97.22	97.22	98.61	97.22	97.22	97.22
Cuadrado Sanos	100	100	100	100	100	95.83
Cuadrado DCL Estable	100	100	100	100	100	100
Cuadrado DCL Evolución	100	100	100	100	100	100
Cubo sanos	97.62	95.24	92.86	95.24	97.62	95.24
Cubo DCL Estable	66.67	73.81	83.33	66.67	73.81	69.05
Cubo DCL Evolución	84.52	86.91	79.76	82.14	86.91	86.91
Minimental Sanos	92.43	93.94	95.46	93.94	95.46	93.94
Minimental DCL Estable	96.18	98.18	96.18	100	98.18	100
Minimental DCL Evolución	96.97	96.97	84.85	96.97	93.94	90.91
Picos-mesetas Sanos	72.44	71.8	78.21	61.54	71.8	59.62
Picos-mesetas DCL Estable	47.72	50.28	53.41	47.72	48.31	49.72
Picos-mesetas DCL Evolución	43.99	40.14	43.13	39.5	43.35	37.6
Triángulos Sanos	100	100	94.45	100	100	100
Triángulos DCL Estable	100	100	100	100	100	100
Triángulos DCL Evolución	94.45	94.45	94.45	94.45	94.45	94.45
Promedio (%)	**80.03**	**79.86**	**81.09**	**77.98**	**80.04**	**77.85**

Tab. 7: Resultados de la evaluación de las asignaciones de GED para la aproximación con RDP combinando varias matrices de matriz de coste

4.3.2.1. Mejores configuraciones

- K: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM y Vector_Suma.
- A: Matriz de coste de sustitución - atributo de Coordenadas.
- M: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM, Vector_Suma, Ángulos_Nodo y Enlaces
- I: Matriz de coste de sustitución - atributo de Coordenadas, Vector_CM y Ángulos_Nodo.

4.3.3. Conclusiones

En la siguiente tabla resumo los promedios totales para todas las configuraciones.

Aproximación	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Esquinas Harris	80.41	18.99	21.74	75.58	45.57	64.28	14.27	18.98
RDP	80.96	23.69	23.66	78.09	50.48	50.72	16.01	25.86

Aproximación	**I**	**J**	**K**	**L**	**M**	**N**
Esquinas Harris	83.93	84.08	81.11	82.73	83.96	81.23
RDP	80.03	79.86	81.09	77.98	80.04	77.85

Tab. 8: Comparativa de la evaluación de las asignaciones de GED para la aproximación con esquinas Harris y RDP.

En general ambas aproximaciones de contorno has conseguido resultados similares (en el entorno del 80 % para las mejores configuraciones). En las configuraciones con únicamente un atributo (A .. H) sólo las coordenadas y el Vector_CM han conseguido resultados similares al de los atributos compuesto para ambas aproximaciones. Por lo que se puede inferir que tiene gran peso en GED.

Otro punto importante es que la combinación de atributos parece mejorar el resultados de las asignaciones con respecto a utilizar solo un atributo. Pero también hay que reseñar que las combinaciones compuestas con las mejores individuales no deriva siempre en una

configuración compuesta mayor. Por ejemplo, las configuraciones I y J de las esquinas Harris se diferencian en un único atributo (Ángulos_Nodo en I y Vector_Suma en K) y teniendo en cuenta que Ángulos_Nodo asigna algo mejor que Vector_Suma (64.28 frente a 45.57); si se observa que la configuración I asigna ligeramente mejor que la la K. Pero esto mismo no se produce para la L y M, en dónde se diferencia en el atributo Degree y enlaces respectivamente (asignando individualmente mejor Degree que Enlaces (21.74 y 14.27) pero las asignaciones de M es ligeramente menor a la de L (83.96 y 82.73).

Hay una particularidad interesante y es que en general las configuraciones individuales aproximadas con RDP son algo mejores que las de Harris y viceversa cuando las configuraciones son compuestas.

Ejemplos de las asignaciones. Por ejemplo en la figura 42 vemos algunas asignaciones adecuadas (grafos modelos arriba y grafos de pacientes abajo). En rojo se indican los nodos de g1 que han sido eliminados y en verde los nodos que se han insertado en g2. Por ejemplo, en **A, B,** y **D** los vértices de g1 se han asignado bien con los de g2 a pesar de las distorsiones. En **C, E** también se han producido buenas asignaciones a simple vista aunque si hay nodos que no se han asignado adecuadamente. Es interesante destacar que cuando faltan muchas partes del dibujo modelo GED ha eliminado muchos nodos del grafo modelo como podemos ver en **E.**

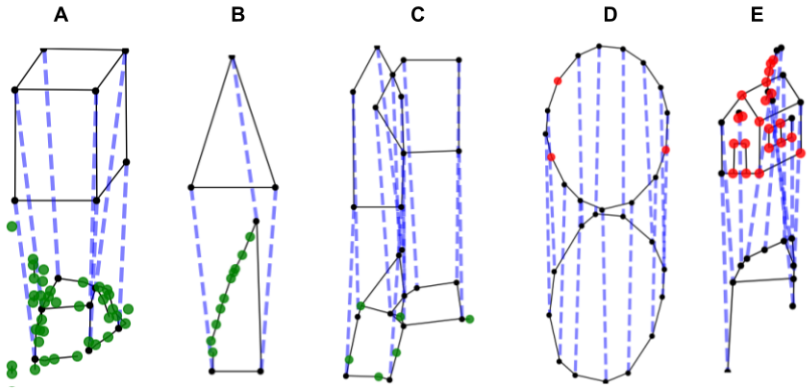


Fig. 42: Ejemplo de asignaciones (1) devueltas por GED (grafos modelos arriba y grafos de pacientes abajo). En rojo se indican los nodos de g1 que han sido eliminados y en verde los nodos que se han insertado en g2.

En la figura 43 vemos más ejemplos de asignaciones adecuadas. Por ejemplo, en **A, B,** hay buenas asignaciones entre nodos. En **C** muestro una comparativa entre una casa asignada perfectamente y otra con algunas deformaciones que a pesar de estas sí se han asignado correctamente la mayoría de los vértices . En cambio, cuando las distorsiones son grandes, como en **D,** la asignación no es tan correcta.

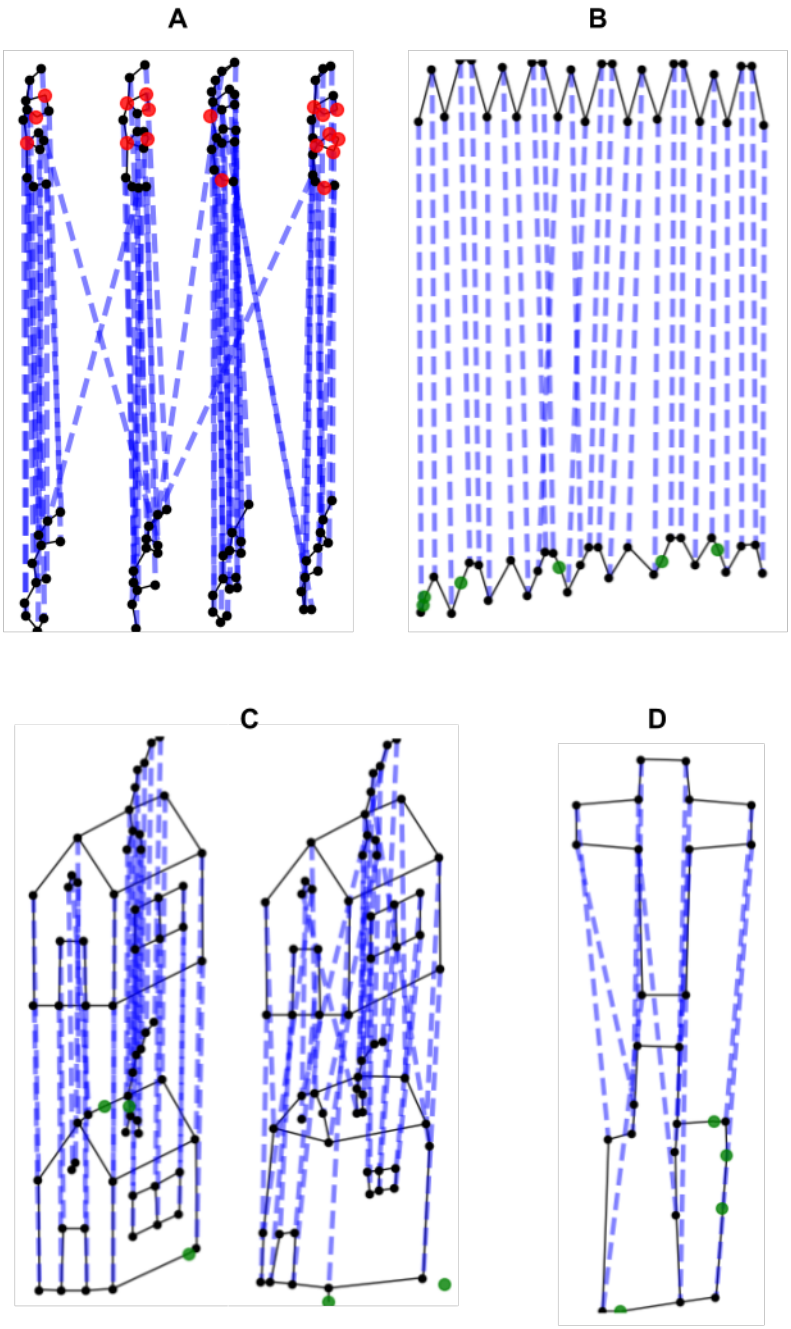


Fig. 43: Ejemplo de asignaciones (2) devueltas por GED (grafos modelos arriba y grafos de pacientes abajo). En rojo se indican los nodos de g1 que han sido eliminados y en verde los nodos que se han insertado en g2.

4.4. Resultados del diagnóstico del DCL

Para evaluar el diagnóstico del DCL se realiza una comparación entre individuos sanos y pacientes con DCL (agrupando en individuos con DCL a los pacientes del dataset catalogados con DCL Estable y DCL Evolución). El diagnóstico del DCL se va a realizar evaluando, con clasificadores, las tres métricas proporcionadas por GED 3.4Recordemos que son:

- Grado de similitud.
- Número de nodos eliminados de g1.
- Número de nodos insertados en g2.

Para realizar las comparaciones se ha utilizado la aplicación Orange [2]. Se han utilizado los siguientes clasificadores con sus parámetros por defecto:

- Naive Bayes
- Tree
- CN2 Rule Induction
- AdaBoost
- kNN
- Neural Network
- Random Forest
- Stochastic Gradient Descent
- SVM
- Logistic Regression

La evaluación se ha efectuado tres veces por cada tipo de figura. Para así determinar con más exactitud el resultado de los clasificadores Vuelvo a hacer dos distinciones en función de la aproximación con esquinas Harris y RDP.

[2] `https://orange.biolab.si/`

Resultados del diagnóstico del DCL para la aproximación con esquinas Harris

En la siguiente tabla se muestran los resultados del diagnóstico del DCL para la aproximación con esquinas Harris. Se ha utilizado la configuración J (atributos de Coordenadas, Vector_CM, Vector_Suma y Ángulos_Nodo). En las columnas de DCL y SANOS se indica el porcentaje de cada clase (sobre el total de instancias) clasificado correctamente.

FIGURA	DCL (% aciertos)	SANOS (% aciertos)	Mejor clasificador
Bucles	75.6	52.4	KNN
	75.6	52.4	KNN
	71.1	55.6	KNN
Casas	67.9	63.5	Tree
	67.9	63.5	Tree
	69.2	60.3	Random Forest
Círculos	76.4	54	Naive Bayes
	76.4	54	Naive Bayes
	76.4	54	Naive Bayes
Cruz	69.3	49.2	Naive Bayes
	69.3	49.2	Naive Bayes
	69.3	49.2	Naive Bayes
Cuadrado	66.3	58.7	CN2 rule inducer
	66.3	58.7	CN2 rule inducer
	66.3	58.7	CN2 rule inducer
Cubo	71.8	71.4	kNN
	71.8	71.4	kNN
	71.8	71.4	CN2 rule inducer
Minimental	67.2	66.7	CN2 rule inducer
	67.2	66.7	SGD
	71.6	60	SGD
Picos-mesetas	60	55.6	Naive Bayes
	60	55.6	Random forest
	60	55.6	Naive Bayes
Triángulos	80.7	34.9	Tree
	80.7	34.9	Tree
	80.7	34.9	Tree

Tab. 9: Diagnóstico DCL para la aproximación con esquinas Harris

Resultados del diagnóstico del DCL para la aproximación con RDP

En la siguiente tabla se muestran los resultados del diagnóstico del DCL para la aproximación con RDP. Se ha utilizado la configuración K (atributos de Coordenadas, Vector_CM y Vector_Suma). En las columnas de DCL y SANOS se indica el porcentaje de cada clase (sobre el total de instancias) clasificado correctamente.

FIGURA	DCL (% aciertos)	SANOS (% aciertos)	Mejor clasificador
Bucles	73.3	55.6	KNN
	73.3	55.6	KNN
	73.3	55.6	KNN
Casas	70.5	60.3	Tree
	70.5	60.3	Tree
	64.1	65.1	Random Forest
Círculos	60.7	63.5	Naive Bayes
	60.7	63.5	Naive Bayes
	60.7	63.5	Naive Bayes
Cruz	73.9	52.4	Naive Bayes
	73.9	52.4	Naive Bayes
	73.9	52.4	Naive Bayes
Cuadrado	70.8	55.6	CN2 rule inducer
	70.8	55.6	CN2 rule inducer
	70.8	55.6	CN2 rule inducer
Cubo	69.2	63.5	kNN
	69.2	63.5	kNN
	64.1	63.5	CN2 rule inducer
Minimental	61.2	63.3	CN2 rule inducer
	59.7	65	SGD
	61.2	65	SGD
Picos-mesetas	55	57.1	Naive Bayes
	65	50.8	Random forest
	55	57.1	Naive Bayes
Triángulos	69.3	50.8	Tree
	69.3	50.8	Tree
	63.6	54	Adaboost

Tab. 10: Diagnóstico DCL para la aproximación con RDP

Resumen de los resultados del diagnóstico DCL

Seguidamente expongo dos gráficas con los resultados de las dos tablas anteriores. En ellas se muestra el porcentaje medio de los aciertos de las clases DCL y SANOS por cada tipo de figura.

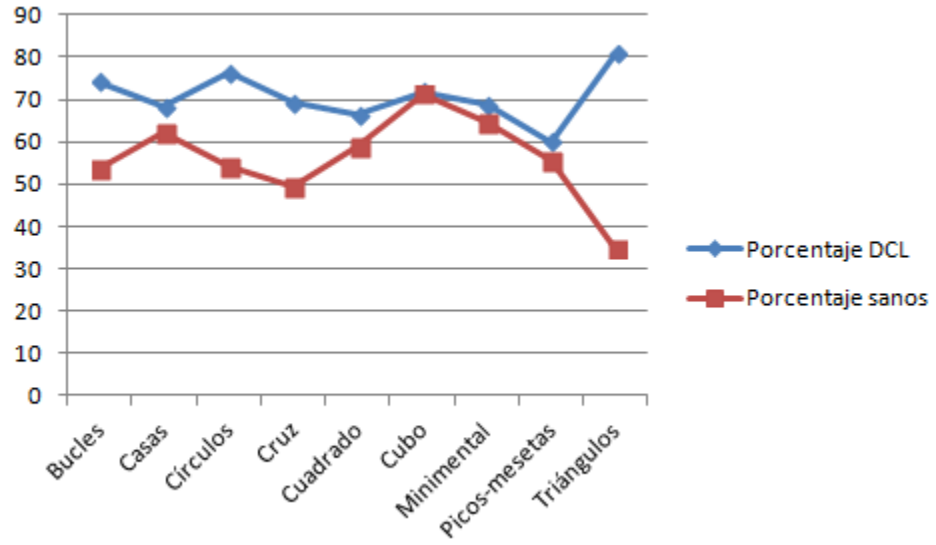


Fig. 44: Gráfica con los resultados del diagnóstico para la aproximación por esquinas Harris

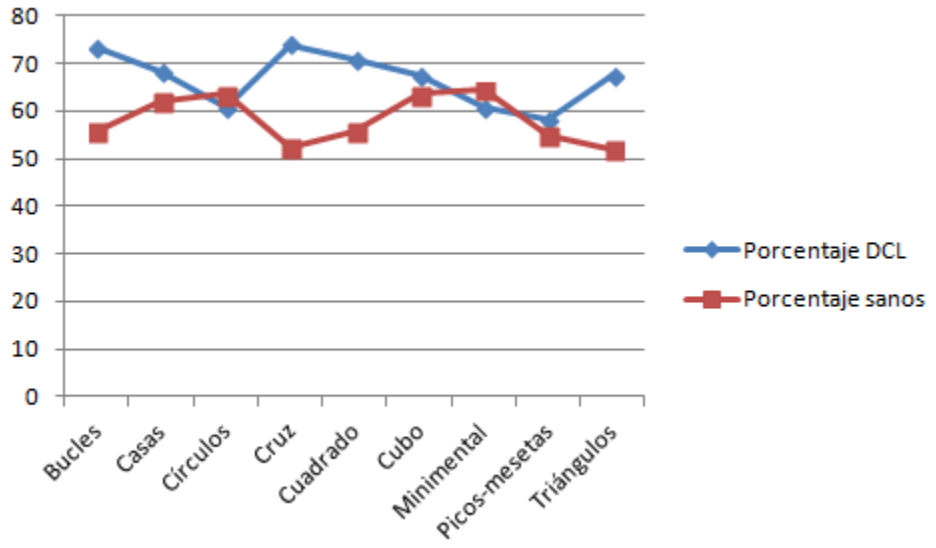


Fig. 45: Gráfica con los resultados del diagnóstico para la aproximación por RDP

5. Conclusiones

5.1. Respuestas a las hipótesis

1. **Cuanto mayor sean las diferencias mayor será el DCL**. Partiendo de los resultados obtenidos en el diagnóstico DCL no es posible corroborar la hipótesis. En la mayoría de las figuras la clasificación para los pacientes sanos ronda el 55 % y 58 % con Harris y RDP respectivamente. Y de un 70 % y 67 % con Harris y RDP respectivamente para el diagnóstico de pacientes con DCL. Debido a que para los pacientes sanos el porcentaje en ambos es inferior al 60 % y a que unicamente hay dos clases (la probabilidad de acertar aleatoriamente es del 50 %) por lo que no se puede hacer un diagnóstico del DCL ni validar la hipótesis.

2. **Cuanto más se asemeja el modelo al dibujo de partida, mejor será el diagnóstico del DCL.** Los resultados del diagnóstico DCL son similares para las dos aproximaciones de contornos. Pero no se puede validar la hipótesis debido al escaso poder discriminatorio de los resultados. Además hay que destacar el resultado de las clasificaciones para los triángulos en la aproximación por Harris. En este caso el mejor resultado para los pacientes sanos es inferior al 40 % y eso teniendo en cuenta que el de las asignaciones es cercano al 100 %. Por lo que se puede concluir que el proceso de GED no es adecuado para calcular la semejanza entre el patrón y el dibujo del paciente.

3. **Para que la asignación obtenida del graph matching sea verosímil hay que evaluar las correspondencias generadas a través de una comprobación supervisada.** Ambas aproximaciones muestran resultados similares (entorno al 80 %) en el diagnóstico del DCL y justo se han obtenido mejores resultados con la de RDP (0.74 frente a 0.69 en el coeficiente de DICE). Por lo que estos resultados si podrían corroborar la hipótesis pero de nuevo los bajos porcentajes de aciertos (especialmente con los sanos) anulan la respuesta afirmativa.

5.2. Conclusiones finales

GED ha conseguido buenas asignaciones (del entorno del 80 %) pero el proceso de GED no es adecuado para calcular la semejanza entre el patrón y el dibujo del paciente visto el resultado del diagnóstico del DCL. En estudios futuros es necesario aproximar el problema de graph matching con otras técnicas como por ejemplo graph embbeding o con otras implementaciones de GED que tengan en cuenta a la vez las operaciones de edición de nodos y enlaces. En vez de manera independiente como se ha expuesto en este trabajo. Otra manera de mejorar el diagnóstico del DCL sería añadir más métricas. Por ejemplo sería interesante combinar la métricas de este trabajo con las métricas globales de los trabajos previos [9].

Referencias bibliográficas

[1] M.D. Claver-Martín (2008) Instrumentos de valoración en el deterioro cognitivo leve. Madrid: Ayuntamiento de Madrid recuperado de: https://www.viguera.com/sepg/pdf/revista/0000/0000_09_16.pdf

[2] Lobo, A., Ezquerra, J., Gómez, F., Sala, J. M. and Seva A. (1979). El mini-examen cognoscitivo. Un test sencillo, práctico, para detectar alteraciones intelectivas en pacientes médicos. Actas Luso Españolas de Neurolología, Psiquiatría y Ciencias Afines, 3, 189-202

[3] Peña-Casanova, J. (1991). Programa integrado de exploración neuropsicológica «test Barcelona». Normalidad, semiología y patología neuropsicológica. Barcelona: Masso

[4] Rey, A. (2003). Rey. Test de copia y de reproducción de memoria de figuras geométricas complejas. Madrid: TEA.

[5] Urbina, S. (2007). Claves para la evaluación con tests psicológicos (A. S. Kaufman & N. L. Kaufman, trad.). Madrid: TEA Ediciones (publicado originalmente en 2004).

[6] Guerrero, J.M., Martínez-Tomás, R., and Peraita, H. (2011). Bayesian Network Based Model for the Diagnosis of Deterioration of Semantic Content Compatible with Alzheimer's Disease. Foundations on Natural and Artificial Computation. Lecture Notes on Computer Science, vol 6686, pp.461-470, Springer, 2011, ISBN: 978-3-642-21343-4.

[7] Guerrero, J.M., Martínez-Tomás, R., Rincón, M., and Peraita, H. (2015). Bayesian network model to support diagnosis of cognitive impairment compatible with an early diagnosis of Alzheimer's disease. Methods of information in medicine.

[8] M. Rincón, S. García-Herranz, M. C. Díaz-Mardomingo, R. Martínez-Tomás, H. Peraita. Automatic Drawing Analysis of Figures Included in Neuropsychological Tests for the Assessment and Diagnosis of Mild Cognitive Impairment. Artificial Computation in Biology and Medicine. Volume 9107 of the series Lecture Notes in Computer Science pp 508-515.

[9] Mikel Val Calvo, Patricia Mayo Tejedor, Roberto Cerrillo Ayuso. Evaluación automática de dibujos realizados respecto a un patrón para la detección de problemas cognitivos. Trabajo final en grupo de la asignatura Visión Artificial del curso 2017/2018.

[10] Tomoaki Shigemori, Zainab Harbi, Hiroharu Kawanaka. Feature Extraction Method for Clock Drawing Test. 19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems. http://dx.doi.org/10.1016/j.procs.2015.08.280

[11] R. O. Canham, S. L. Smith and A. M. Tyrrell. Recognition and Grading of Distorted Geometric Shapes Complex Figure. Article in Formal Pattern Analysis & Applications 3(4):335-347 · December 2000. DOI: 10.1007/s100440070005.

[12] R.M. Guest, M.C. Fairhurst, J.M. Potter. Automated extraction of image segments from clinically diagnostic hand-drawn geometric shapes. Published in: Euromicro Conference, 2000. Proceedings of the 26th.

[13] Chatbri, Houssem & Kameyama, Keisuke & Kwan, Paul & Little, Suzanne & O'Connor, Noel. (2018). A novel shape descriptor based on salient keypoints detection for binary image matching and retrieval. Multimedia Tools and Applications. 10.1007/s11042-018-6054-x.

[14] M. Kopczynski, M. Sester Representation of sketch data for localisation in large data sets. (2004).

[15] TA, Anh Phuong. Inexact graph matching techniques: Application to object detection and human action recognition. 2010. Tesis Doctoral. Lyon, INSA.

[16] Zhang, and Lu. "Review of Shape Representation and Description Techniques." Pattern Recognition 37.1 (2004): 1-19. Web.

[17] Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. Pattern Analysis and Machine Intelligence, IEEE Transactions on 24(4), 509{522 (2002)

[18] Opelt A., Pinz A., Zisserman A. (2006) A Boundary-Fragment-Model for Object Detection. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3952. Springer, Berlin, Heidelberg

[19] Ferrari V., Fevrier L., Jurie F., Schmid C.: Groups of adjacent contour segments for object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 1 (2008), 36–51.

[20] Ferrari, V, Tuytelaars, Tinne, and Van Gool, Luc. "Object Detection by Contour Segment Networks." Lecture Notes in Computer Science 3953 (2006): 14-28. Web.

[21] Sadegh Abbasi, Farzin Mokhtarian, Josef Kittler: Curvature Scale Space Image in Shape Similarity Retrieval. Multimedia Syst. 7(6): 467-476 (1999).

[22] (12 de abril, 2013). CSS – Curvature Scale Space in OpenCV – Vision & Graphics Group. Recuperado de `https://vgg.fiit.stuba.sk/2013-04/css`

[23] Arkin, E. M., Chew, L. P., Huttenlocher, D. P., Kedem, K., & Mitchell, J. S. (1989). An efficiently computable metric for comparing polygonal shapes (No. CU-CSD-TR-89-1007). CORNELL UNIV ITHACA NY DEPT OF COMPUTER SCIENCE.

[24] Elliott, Hilary. 1 An Efficiently Computable Metric for Comparing Polygonal Shapes Esther M. Arkin, L. Paul Chew, Daniel P. Huttenlocher, Klara Kedem, Joseph S.B. Mitchell. [ppt] Recuperado de `https://slideplayer.com/slide/4700280/`

[25] Lowe D. G.: Object recognition from local scale-invariant features. In ICCV (1999), pp. 1150–1157.

[26] Harris C., Stephens M.: A combined corner and edge detector. In Proceedings of the 4th Alvey Vision Conference (1988), pp. 147–151.

[27] Bay H., Tuytelaars T., Van Gool L. (2006) SURF: Speeded Up Robust Features. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg

[28] Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011: 2564-2571.

[29] Dalal N., Triggs B., Schmid C. (2006) Human Detection Using Oriented Histograms of Flow and Appearance. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3952. Springer, Berlin, Heidelberg

[30] (3 de septiembre, 2019). ORB feature detector and binary descriptor [Imagen]. Recuperado de `https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_orb.html`

[31] WuX., Bhanu B.: Gabor wavelets for 3-d object recognition. In ICCV '95: Proceedings of the Fifth International Conference on Computer Vision (Washington, DC, USA, 1995), IEEE Computer Society, p. 537.

[32] TA, Anh Phuong. Inexact graph matching techniques: Application to object detection and human action recognition. 2010. Tesis Doctoral. Lyon, INSA.

[33] Horst Bunke. Recent Developments in Graph Matching. Published in: Pattern Recognition, 2000. Proceedings. 15th International Conference on Pattern Recognition (ICPR'00). DOI: 10.1109/ICPR.2000.906030.

[34] J. Ullman. An algorithm for subgraph isomorphism. Journal of the Association for Computing Machinery, 23(1):31–42, 1976.

[35] Endika Bengoetxea, Pedro Larrañaga, Isabelle Bloch, Aymeric Perchant, Claudia Boeres. Inexact graph matching by means of estimation of distribution algorithms. Published by Elsevier Science Ltd. PII: S0031-3203(01)00232-1.

[36] E. Bengoetxea, P. Larrañnaga, I. Bloch et al. Inexact graph matching using learning and simulation of Bayesian networks. An empirical comparison between different approaches with synthetic data. In Proceedings of CaNew workshop, ECAI 2000 Conference, ECCAI. Berlin, aug 2000.

[37] Riesen, Kaspar, and Horst Bunke. "Approximate graph edit distance computation by means of bipartite graph matching." Image and Vision computing 27.7 (2009): 950-959.

[38] [2] Fischer, Andreas, et al. "Approximation of graph edit distance based on Hausdorff matching." Pattern recognition 48.2 (2015): 331-343.

[39] P. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions of Systems, Science, and Cybernetics 4 (2) (1968) 100–107.

[40] Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5, 32–38 (1957)

[41] Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9 (1): 62–66. doi:10.1109/TSMC.1979.4310076

[42] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing, 1(3):244 – 256, 1972.

[43] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization, 10(2):112–122, 1973.

[44] T.-C. Lee, R.L. Kashyap and C.-N. Chu, Building skeleton models via 3-D medial surface/axis thinning algorithms. Computer Vision, Graphics, and Image Processing, 56(6):462-478, 1994.

[45] (28 de junio de 2018) Pau Riba y Anjan Dutta. Recuperado de `https://github.com/priba/aproximated_ged`